Principles (and Philosophy) of Linear Algebra
A Gentle Introduction

Paul Mayer

December 2023

1 Introduction

This text serves as a gentle motivating introduction to some of the principles (and philosophy) behind linear algebra. This is aimed at undergraduate students taking a linear algebra class - in particular engineering students who are expected to understand and use linear algebra to build stuff, however it may also prove helpful for philosophy majors and anyone else interested in the ideas behind linear algebra. Linear Algebra is an extremely important field that extends everyday concepts about geometry and algebra into higher spaces. Linear algebra is used extensively in statistics, mechanical engineering, circuit theory, signal processing, economics and data science/machine learning. The difficulty learning linear algebra is that it can be very abstract and hard to visualize (since we are often working in spaces without a fixed dimension). At bottom, Linear Algebra is about abstract ideas that extend beyond 2-D and 3-D, and is much more than simply doing operations on matrices (after all, why do we care about matrices?). Definitions are important, but I find students tend to struggle the most with motivating the *why* behind each of these things. This can be difficult for professors to understand because we are so used to using the language of linear algebra we take for granted why these questions are important in the first place. In other words, we may understand the "why" (after years of using these tools), but our students may not. This text seeks to explain this "why" in an accessable, engaging way.

2 What is Linearity?

To talk about Linear Algebra, we need to understand what linearity is (and what algebra is). Algebra is the easier one to define - algebra involves the use of abstract symbols instead of numbers. Why is this helpful? The principle behind algebra is abstraction - a word that means being able to think at a higher (or more abstract) level that removes details. I like to think of abstraction as enabling laziness: rather than think about little details, we can "abstract" them away and think about things more generally. So algebra involves thinking about things in a general (and lazy) way - rather than having to solve an equation a

bunch of different times as things change, we can solve it in terms of a variable and substitute it in for a number when we need to. We can use this general form multiple times as needed - often allowing us to do less work.

Linearity involves the word "line," so a first thought is that linearity (and this linear algebra) involves lines. This is partially true, but it does not tell the whole story. Linearity is a concept that involves two definitions that are so fundamental they are worth memorizing - we will go into more detail to the "why" later [1]. A function f is defined as **linear** if it satisfies the following two properties:

1. Superposition: $f(x+y) = f(x) + f(y)$ and
2. Homogeneity: $f(ax) = af(x)$

We have listed these properties with pretty much no context, so let us go into more detail on each. Superposition is the idea that you get the same result whether you addition before or after applying f. Integrals are like this, which makes them easy to break apart and recombine. Plenty of physical properties also exhibit this behavior: consider weighing items on a scale - if you weigh them separately and add the results, this is the same as weighing them all at the same time. However many systems in the world do not exhibit homogeneity. Consider the square function $f(x) = x^2$: $(x+y)^2$ is not the same as $x^2 + y^2$. Also, putting eggs in the oven and dough in the oven and then combining the result is not the same as putting dough and eggs into the oven at the same time. So baking, for the most part, is nonlinear because it does not exhibit superposition (recall that to be linear, it needs to satisfy both superposition and homogeneity).

Homogeneity is another way of saying scalar invariance - in other words, if an input gives a specific output, then a scaled version of the input gives a scaled version of the output (with the same scalar number). This is the behavior that is the most "line-like." Standard addition behaves like this. Weighing items does as well - if I put two times the number of grain on the scale, I will get twice the weight. The square function fails here again, as $(ax)^2 \neq ax^2$. Sin and Cos also fail, since $\sin(x+y) \neq \sin(x) + \sin(y)$. Finally, consider microwaving items like hot pockets. If putting a single pocket in the microwave gives me a (hot) hot pocket in 7 minutes, then putting two pockets in the microwave should give me two hot pockets in 7 minutes. If the box of hot pockets tells me heating two at a time takes 10 minutes, while heating one only takes 7, then heating hot pockets is nonlinear.

In general, we show a system is linear by starting with $f(x+y)$ and showing it leads to $f(x)+f(y)$ (establishing superposition), and starting with $f(ax)$ and showing this leads to $af(x)$ (establishing homogeneity). To show systems are not linear (nonlinear), we just need to provide a single counterexample where either superposition or homogeneity fails. As a result, showing a nonlinear system is not linear is often easier than showing a linear system is linear.

[1] I should note that Linear Algebra is very concept heavy, which means definitions are very important for understanding these concepts. Memorizing these will be helpful

Hopefully you now understand what superposition and homogeneity are. What may not be clear is why we care about these properties enough to make a whole field out of them: after all, many systems we care about (and are simple) are not linear, like the square function or an oven. The reason we care about linearity is that these assumptions are about as easy and simple as we can make things to solve problems with them. When we look to nonlinear systems, much of the intuition we have gain from linear systems will help. But honestly, the main reason we use linear systems theory is that nonlinear systems are much harder to solve. For all the difficulty found in solving linear systems (and linear algebra is not an easy class), nonlinear systems are often way more difficult if not completely intractable. However not all is lost - learning about linear systems and linear algebra can help us even when the underlying dynamics are nonlinear. For instance:

- Often at certain scales, a nonlinear system will behave somewhat linearly (this is often called a "linear region" of device), so linear algebra will help here.
- If we want to be lazy, we can simply assume the system behaves linearly. In some cases, the difference is negligible or a linear approximation is "good enough" to help us. In engineering, a linear approximation may be enough for us to build awesome stuff without getting too bogged down by details. For example, whenever you see physicists use the approximation $\sin(x) \approx x$ (for small x,) they are using a linear approximation for small x.
- We can often perform a transformation to use linear systems theory on a nonlinear system. What we do is perform some sort of transformation, use linear algebra to solve it, and then perform a reverse transformation back when we are done. This is how we can use linear algebra to solve for polynomial interpolation (i.e. curve fitting) for higher order polynomials.

3 From numbers to vectors

We use numbers for a variety of purposes, including ordering and labeling ("I am first in line," and "I'll have the number 1 combo"), measuring ("I am 4.2 miles away") and counting ("I have 4 beers in my fridge"). There are different kinds of numbers, including integers ($\mathbb{Z}$), rationals ($\mathbb{Q}$), real numbers ($\mathbb{R}$), and complex numbers ($\mathbb{C}$). The most common we use in Linear Algebra are real numbers, as they form a *continuum* (meaning there are no "holes" in between numbers). To say x refers to a real number, we write $x \in \mathbb{R}$, which is read "the variable x is an element of the real numbers".

However, there are times we want to represent something that cannot be captured by a single number. One example of this are points in $2D$ space (try seeing if you can represent an arbitrary location on a plane with a single number). Or suppose you want to represent the number of dogs and cats a

person has [2]. To represent these quantities, we use what are called **vectors**, which is just an ordered list of numbers. The ordered part is important because the *location* of each number matters: 5 dogs and 2 cats is different than 2 dogs and 5 cats, and 4 units up, 3 to the right is different than 3 units up, 4 to the right.

To represent a vector of a given type, we use the superscript to denote the *dimension* of the vector, which corresponds to the elements (or numbers) in the each vector. So $\vec{x} \in \mathbb{R}^2$ refers to a vector which contains two real numbers, while $\vec{z} \in \mathbb{Z}^4$ refers to a vector with four integers. To try and avoid confusion, I put a little arrow on top of the variables $\vec{x}$ and $\vec{z}$ to show they refer to vectors (and not a single number). There are other conventions used to represent vectors you may see: physicists may use one line on top, like this $\overline{x}$, and sometimes Electrical Engineers and Mathematicians use lowercase letters that are bolded to represent vectors, like this $\mathbf{x}$, so be aware of the fact that the notation may change depending on the literature you are reading. Luckily, most people notate when a variable refers to a vector by writing something like $\mathbf{x} \in \mathbb{R}^4$.

4 Dimension

The **dimension** of a vector is the number of elements (numbers) it has. Here is where the "algebra" in linear algebra comes in: we often use a positive integer (the most common being n) to represent the dimension, so we can generalize our results to any dimension. So when we say $\vec{x} \in \mathbb{R}^n$, our vector $\vec{x}$ refers to an $n-$dimensional vector of real numbers – where n is a variable. This allows us to prove results that exist independently of a specific dimension. Why is this helpful? If we can prove a result in $\mathbb{R}^n$, we have a proof that works for $2D$, $3D$, and any other "D" we like (D meaning dimension) – meaning more results for less work! Furthermore, our proofs are often easier when we abstract away from a given dimension, especially if the dimension in question is very large and the proof would involve a lot of tedious comparisons. When it comes to Cartesian spaces of dimension n (meaning spaces that behave like 2D and 3D grids, but of arbitrary dimension) we need vectors of length n to specify each point in the space – any more and we have unneeded redundancy, and any less and we will not be able to access every point in the space. This will be formalized later, but for now just remember each coordinate needs its own number or "cell."

It can help to think of a vector as a container or list of individual numbers, where the dimension is the length of the list/number of members. This is particularly helpful when representing vectors within a computer, since each number in the vector will have a specific location in memory that can be accessed as needed. In general, we *index* into a vector at a given location using a subscript

[2] One creative way to try and do this is to use a decimal point, so someone with 5 cats and 2 dogs could be written as 5.2. This doesn't quite work though, as the real number 5.2 is larger than 2.9, despite the latter person having more dogs. This is a bit of a cheat because we are simply writing two different numbers with a symbol to separate them - the encoding of the numbers are separate and it would arguably be easier to just use two separate numbers in this case

representing where we are "peeking." For instance, consider the vector $\vec{x} = [-2, 1000, 0.3, 42]^T \in \mathbb{R}^4$. We can grab the 2nd component of the vector by saying $\vec{x}_2 = 1000$. Alternatively, we can index into the vector using a variable, such as when adding the individual components together to get the sum of the elements, such as

$$\sum_{i=1}^{n} \vec{x}_i = -2 + 1000 + 0.3 + 42 = 1040.3$$

A vast majority of the time, we let n be a positive integer [3], however there are times where a mathematical object can be described where n refers to a rational number instead. In cases like these, such as $\mathbb{R}^{3/2}$, we have a fractional dimension, or fractal, for short. You won't encounter them much in introductory linear algebra, but they do look cool!

5 Points and Vectors

One of the basic building blocks of a geometric space are things called points - mathematical objects that exist at a specific location. Many of you are familiar with a standard 2-D Cartesian grid of (x, y) pairs. In 2-D, each (x, y) pair refers to a specific geometric point on a 2D plane. Linear Algebra extends this definition from 2-D to arbitrary dimensions that are impossible to visualize, however the idea is the same: the number in each space refers to how many units to move in a specified direction.

Points and vectors are actually different things, however we often use the words interchangeably because the difference between them is often too subtle to really matter, and I am not a stickler about this difference because in practice it often does not matter. In general, a point refers to the geometric object *at* a given location in space, while a vector refers to a displacement in space which has a direction and a magnitude. When this displacement is from the origin (i.e. the "starting point," where zero is defined), vector "points" exactly to the point in question. In other words, vectors are relative to some starting point, while points refer to absolute values in space. When vectors are specified relative to the origin, they correspond with points (in the sense that they "point to" the point specified by its coordinates).

There is quite a bit more that can be said about the difference between points and vectors, but much of it is unimportant for our current discussion. With that being said, linear algebra works with vector spaces, not necessarily *points*, so we can do linear algebra on things that have no explicit geometry. However we often use standard Euclidean geometry to *visualize* vectors and vector spaces, which forms a helpful analogy between the 2-D and 3-D plots we are familiar with and more abstract spaces. For instance, I can consider vectors of the form $[b, g]^T$ which represent the number of boy (b) and girl (g) children a given person has: this does not necessarily *have* any geometry since it is simply

[3] It is common mathematical convention for the variable n to refer to a positive integer

an encoding of things in the world, however we can *use* geometry to visualize and compare these vectors.

6 Vector Spaces, Addition, and Multiplication

Vector spaces are the main thing we work with in linear algebra. A Vector Space X involves the following:

- A (nonempty) set of vectors S
- A field [4] of scalars F (usually the real numbers, $\mathbb{R}$)
- Two operations, $+$ and $*$, that behave the way you expect

We sometimes write the vector space as $X = (S, F, +, *)$ to specify each of the four things needed, but since addition and scalar multiplication behave "normally," we often just omit writing them out of laziness. To be a legitimate vector space, the vector space needs to be *closed* under vector addition and scalar multiplication.

What does this mean? To be closed under vector addition means if I take two vectors $\vec{x}, \vec{y} \in X$ (this is shorthand for saying $\vec{x}$ and $\vec{y}$ are elements of the set X), the *vector sum*, $\vec{x} + \vec{y}$ is also in X. Written in terms of shorthand, we say $\vec{x}, \vec{y} \in X \implies \vec{x} + \vec{y} \in X$, which basically reads as "if $\vec{x}$ and $\vec{y}$ are vectors in a vector space X, then their sum $\vec{x} + \vec{y}$ must also be in X". What happens if I find a pair of vectors in a given set whose vector sum is not in the set? Then the set in question is not a legitimate vector space – sorry. What this captures is the idea that you cannot "add your way out" of the space: any pair of vectors can be added together and their sum will be in the space. You can take this sum, which itself is a vector, and add it to any vectors in the space, and *that* vector will also be in the space. In other words, there are no illegitimate children in a legitimate vector space. Vector addition is done by simply adding together the elements in each cell, $[x_1, x_2, \ldots, x_n]^T + [y_1, y_2, \ldots, y_n] = [x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n]$.

Closure under scalar multiplication means if you take a vector, squish or stretch it out (this is what scalar multiplication does), then this squished or stretched vector will also be in the space. Recall that our vector space includes an associated field F. So if I take some scalar (we use the word scalar to differentiate it from a vector, since a scalar is a single number whereas a vector can be a collection of numbers), and multiply it by any vector in the space, this result is also in the space. When multiplying a vector by a scalar, you just multiply each component (i.e. number) of the vector by the same scalar, so $2 \cdot [3, 4]^T = [6, 8]^T$. Closure under scalar multiplication can be written concisely as $\vec{x} \in X, a \in F \implies a\vec{x} \in F$.

Multiplication is defined for scalars and vectors, but what about multiplication between two vectors? This is actually *undefined* (which means you can

[4] A field is a set of individual numbers (not vectors) where addition, subtraction, multiplication, and division work in the usual way

define it however you want, but it is not needed for a vector space, so we often do not define it any particular way). Recall that to have a legitimate vector space we only need *vector* addition (addition between vectors), and *scalar* multiplication (multiplication between a scalar and a vector) – we do not need vector multiplication defined. So you are free to invent it and define it however you like: after all + and $*$ are just symbols that reference some operation that gives a result according to some defined rule. More often than not, however, we just leave it alone.

7 Orthogonality

Orthogonality is defined as a generalization of perpendicularity (i.e. things being at right angles). However angles being 90 degrees is not really what we care about. The principle of orthogonality is simply that of independence - it just so happens that in Euclidean (i.e. normal) spaces, independence means at 90 degree angles. Orthogonality can generalize beyond Euclidean spaces to all sorts of weird math areas, but it also applies to our everyday life.

Consider the following example: suppose I want to make a my favorite coffee which has 3 creams and 2 sugars. For simplicity, let's assume a single cream has 1 unit of creaminess and a single packet of sugar has 1 unit of sweetness. If creaminess and sweetness are orthogonal (i.e. independent) to each other, cream only affects creaminess and sugar only affects sweetness. Sugar and Cream are *orthogonal* to each other because adding sugar does not affect the creaminess and adding cream does not affect the sweetness. We can draw this on a Cartesian grid with the x axis representing creaminess and the y axis representing sweetness: the vector $(3, 2)$ represents my favorite kind of coffee.

Now, suppose I run out of cream, but I do have sweet cream instead. A single sweet cream has 1 unit of creaminess and 0.5 units of sweetness. Sweet cream and sugar are *not* independent of each other, since sweet cream and sugar both affect the sweetness of my coffee. So to make my optimal coffee, I now need to do more work to figure out how much sweet cream and sugar I need to add. This includes taking into account the interaction between sugar and sweet cream - something you do not have to do when having independent (orthogonal) ingredients. Worse still, I have to do this before I have had any coffee.

In general, orthogonality makes things easier and enables us to be lazy, so we often want an orthogonal basis to build stuff or simply understand what is going on. Luckily, the tools of linear algebra allow us to perform an *orthogonalization*, removing the dependence between different variables (such as sweet cream and cream), allowing us work in a space where things don't affect each other. The most well known method of orthogonalization is called the Gram–Schmidt process – we will talk about it later, but in essence it simply subtracts away the dependence each variable has with the others. This method works because we are working in a linear vector space.

8 Inner Products

Recall that vectors tend to have a direction and a magnitude. This is especially clear when we draw them as arrows - when positioned at the origin, the arrow "points" in some specific direction away from the origin at some length. We often want some way of quantifying how similar the directions of these arrows are. This is what the inner product allows us to do – it defines a way of measuring how similar two vectors are that takes into account their magnitude and angle.

You may not care about arrows, but you may want to know whether people like the same movies as you do. For simplicity, let's say we only have action movies and romance movies, and we rate whether we like this type of on a scale where large positive numbers mean you really like a given movie and large negative numbers mean you really dislike this type of movie. My movie tastes may best be described as the vector $\vec{m} = [5, -0.5]^T$, where the first component says I really like action movies and the second component says I somewhat dislike romance movies, (taking into account the few RomComs I enjoy but would not admit to in public).

Now suppose we ask two people what their movie tastes are, person A's "preference vector" is $\vec{a} = [4, -1]^T$ while person B's "preference vector" is $\vec{b} = [-3, 3]$. Which of these people have a more similar movie taste to me? Now you can probably tell by just looking at these numbers that it is person A, but remember we want to formalize these notions so we can do this comparison when we have a bunch of genres (i.e. in a super high dimensional space). The inner product is a way to do this comparison - it multiplies the corresponding values in each component together, and adds the result: if two people rate the same type of movie highly, then you will be adding a very large positive number. If one person rates it negatively and the other positively, then the inner product (you can think of it as a "similarity value") will be negative.

We define the standard inner product [5] between two n-dimensional vectors is defined as follows:

$$\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i \tag{1}$$

. Now let's compare my preferences to Person A and Person B: $\langle \vec{m}, \vec{a} \rangle = 5 \cdot 4 + -0.5 \cdot -1 = 20.5$ and $\langle \vec{m}, \vec{a} \rangle = 5 \cdot -3 + -0.5 \cdot 3 = -16.5$. The fact that $20.5 > -16.5$ tells me that person A has more similar movie tastes to me than person B. At the risk of beating a dead horse, let's try and understand why: both me and person A really like action movies, so we multiply two larger (relative to the others) numbers together to get 20. We also both dislike romantic movies, however since we both dislike them, the negatives cancel out and we end up adding *another* positive number: the inner product is larger because our tastes are in the same "direction" for each component. When we compare me and

[5] Vector Spaces do not need an inner product defined, however it is often helpful to define one. The one defined here is not the only legitimate inner product, but it is by far the most popular and arguably the most useful. My recommendation: when in doubt, use the standard inner product defined here.

person B, the opposite happens: they like movies I do not, and I like movies they do not, which is reflected in the sign of each additive term in the inner product – both are negative.

9 Norms and Normalized Inner Products

9.1 The 2-Norm

The word "norm" simply refers to a measure of length – and it is used to generalize the idea of distance we are all familiar with. There are different kinds of norms, but each must satisfy a certain set of rules [1] to be considered a legitimate norm. The most popular norm we will see is what is called the 2-norm, which (if using the standard inner product above), is simply the squareroot of a vector's inner product with itself:

$$\|x\|_2^2 = \langle \vec{x}, \vec{x} \rangle = \sqrt{\sum_{i=1}^{n} \vec{x}_i^2} \tag{2}$$

It is called the 2-norm because we square each term and take the squareroot. We can use a different power (other than the number 2) to get what is called the p-norm (though when adding the terms, we take the absolute value as well, something we do not have to do when p is even), but the 2-norm is by far the most popular and useful.

In Cartesian spaces, it refers to the idea of distance we are all familiar with (and in 2 dimensions, becomes the Pythagorean theorem: $c^2 = a^2 + b^2$). This is useful because we now have a measure of distance beyond two dimensions: we can calculate distances in 3 dimensions, or even 10 if we like, and it behaves the way we are used to.

9.2 Normalization

A Norm is a measure of distance, and for a vector, refers to its length. Recall that a vector can be viewed as an arrow pointing from the origin to a specific point, so the length or norm of this vector is its distance from the origin to the point it "points to" (in other words, the length of the arrow).

In some cases, we care less about the length of the vector and only care about its direction, and in these cases, we **normalize** the vector which means we divide by its norm. Dividing a vector by its norm will give us a vector in the same direction that has norm 1 (A vector with norm 1 is called a **unit vector**). You can think of the number 1 as a mathematical unit, or you can remember that "unit" sounds kind of like "uno," the Spanish word for one.

Why is this helpful? Well, consider our movie example previously – my vector of movie preferences of action and romance movies is $\vec{m} = [5, -0.5]^T$. Suppose we ask two more people to rate their movie preferences: person C tells us their preferences are $\vec{c} = [5, -0.5]^T$, while person D tells us their preferences

are $d = [1000, 10]^T$. Woah! Person D must really like movies. Anyways, which of these people has more similar movie preferences to me? In theory, it seems person C should – after all, their preferences are *the exact same* as mine, while person D has slightly different tastes when it comes to romance movies. However, when we compute the inner products, we get $\langle \vec{m}, \vec{c} \rangle = 25 + 1 = 26$, while $\langle \vec{m}, \vec{d} \rangle = 5000 - 5 = 4995$. When it comes to inner products, it seems that person D is more similar to me. What is going on?

The issue here is that the norm of Person D's preference vector is really large – when we asked them to pick a "large number" for movies genres they liked, they did what we asked. The problem is that we are not working with the same standard units of movie likiness: I put 5 for movies that I really liked, while person D chose numbers in the thousands. As a result, the inner product is larger (even though are preferences differ) simply because they chose larger numbers. I do not care whether or not someone chooses larger numbers – what I care about is whether the *relationship between* their preferences are similar to mine.

So to solve this problem, we can normalize each vector to a unit vector before we take the inner product – the preference vector will then always be of length 1, but its direction will point to how much they like each genre relative to a sum-of-squares length of 1. Calculating the norms, we get that $\frac{m}{\|m\|_2} = \frac{1}{\sqrt{25+0.25}}[5, -0.5]^T = [0.995, -0.0995]^T$. Likewise, $\frac{\vec{c}}{\|\vec{c}\|_2} = [0.995, -0.0995]^T$ (remember they have the same preference vector) and $\frac{\vec{d}}{\|\vec{d}\|_2} = \frac{1}{\sqrt{1000100}}[1000, 10]^T = [0.999, 0.00999]^T$. Dividing by the norm of each vector really reeled in Person D's preference vector. Now let us compute the inner product with these unit vectors: $\langle \frac{\vec{m}}{\|\vec{m}\|_2}, \frac{\vec{c}}{\|\vec{c}\|_2} \rangle = (0.995)^2 + (-0.0995)^2 = 1$, while $\langle \frac{\vec{m}}{\|\vec{m}\|_2}, \frac{\vec{d}}{\|\vec{d}\|_2} \rangle = 0.995 \cdot 0.999 + -0.0995 \cdot 0.00999 = 0.99875$.

So by using normalized vectors, we have enforced a consistent scale between the vectors, and we now have a scale at which my preference vector to someone else with the same preference vector will be maximum. Enforcing this scale is probably something we should have done when we asked the question to begin with. I should note that normalizing vectors is not always something we want to do – by normalizing vectors, we remove information about the length of the vector that may be useful. In our case, we have removed information about the overall "intensity" of how much they like movies, so someone with a preference vector of $[-0.1, 0.1]$ will have the same normalized preference vector as someone who answers $[-10, 10]$. This may or may not be what we want to do.

How do we know? Well, the answer lies with where the "information" we want to represent or find is stored. If all we care about are the *relative* preferences of each movie, than normalization will actually help us find and compare different preferences more accurately, as it removes any information about the intensity of their preferences. In other words, normalization gives us a "common ground" to compare them. However if we also want to know how *much* they like movies, then normalization will be the wrong thing, as it removes the intensity of their vector. In this case, I believe normalization was the right thing to do,

however we could have avoided this if, when we asked them to pick their preferences, we gave them a "scale" that we want them to be consistent about. For instance, we could say "5 represents a movie genre you love to death, while -5 is a movie genre you hate with a seething passion." This calibrates the scale so the overall intensity has more meaning than leaving it up to peoples' interpretation on what a "large positive number" means.

10 Matrices

If you have been following along in order, you may be surprised that we have not discussed matrices yet. In fact, most linear algebra books (such as Gil Strang's legendary book "Linear Algebra and Its Applications [2]" a book I used, my dad used, my PhD advisor used, and my PhD advisor's dad used to learn linear algebra – seriously) start out talking about matrices in Chapter 1. While matrices are important, I think they are often overemphasized relative to the underlying ideas that motivate them. So let's begin by explaining why we need matrices.

Recall that a vector is an ordered list of numbers, where the *dimension* of the vector is the number of numbers inside it. However there are times we want to represent an ordered list of *vectors*, not just numbers. A *matrix* can be thought of as just a vector of *vectors*. As a result, a matrix has *two* dimensions, one referring to the dimension of each of the vectors inside of it (they must all be the same), and the other referring to the number of vectors that make up the matrix. Think about a spreadsheet or the game of battleship, where you have rows and columns, and each row and column gives the location of a cell which contains some information (like a number or part of a war boat).

At this point, we need to decide whether our vectors make up *rows* in the matrix, lined up top-to-bottom, or *columns*, stacked left-to-right, and unfortunately there is no agreed-upon way to do this. Mathematicians, Electrical Engineers, and Physicists tend to make vectors columns of the matrix, while statisticians tend to make vectors the rows (and I have no idea why there is this difference in convention). For this text I will use column vectors but be aware you may see the opposite depending on which literature you use. In fact, we have been using this notation all along: when we write $\vec{v} = [1, 2, 3]^T$, the superscript "T" tells us to turn this list of numbers from left to right to go from top to bottom (that way it does not take up unnecessary space in the text). This "T" refers to transpose, and I will define this more rigorously later, but for now just know

$$[x_1, x_2, \ldots, x_n]^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n. \end{bmatrix} \tag{3}$$

This will be much easier to see with an example. Let's take movie preferences for action and romance as before: my movie preferences vector was

$\vec{m} = [5, -0.5]^T$, Person A's preference vector $\vec{a} = [4, -1]^T$ while person B's preference vector was $\vec{b} = [-3, 3]^T$. We can create a matrix of the collection of all our movie preferences as follows:

$$P = [\vec{m}, \vec{a}, \vec{b}] = \begin{bmatrix} 5 & 4 & -3 \\ -0.5 & -1 & 3 \end{bmatrix}$$

Note that the term $[\vec{m}, \vec{a}, \vec{b}]$ works because the vectors $\vec{m}, \vec{a}, \vec{b}$ are all **column vectors**, meaning each is a tall and skinny list of numbers from top to bottom. If they were instead **row vectors** (short and fat lists of numbers from left to right), then $[\vec{m}, \vec{a}, \vec{b}]$ would just be a really long vector. Pay attention to the way the individual vectors match up in the matrix – this tends to trip up students (and it tripped me up when I was a linear algebra student as well). Also note that we used a capital letter (P) to refer to the matrix – this is a pretty standard convention I recommend you adopt (some also use a bold capital letter).

Like individual vectors, matrices can be indexed into as well. Convention dictates we specify the row first and the column second, so $P_{1,3}$ accesses the element in the 1st row, 3rd column, which is the number -3 (starting from the top-left, down one and to the right three). Likewise, $P_{2,1} = -0.5$, as that is the number in the second row, first column (from the top-left down two, right one). Remember that the row comes first and the column comes second – this always trips up new students, and I would get confused about this as an undergraduate because I was used to the Cartesian grid where the first number specified a number on the horizontal (x) axis while the second specified a number on the vertical (y) axis. What may help is to remember that we start in the *upper-left* hand corner of the matrix, and go *down* by the first number and then *right* by the second (you go down before going right because "down" comes before "right" in the dictionary – not the best mnemonic so come up with your own if you can think of a better one). It is important you remember this because the dimensions of matrix multiplication, which we will talk about in a minute, are easy to mess up.

11 Matrix-Vector Products and Matrix Multiplication

Remember those problems you used to solve in algebra that gave you a system of linear equations (meaning a set of different, interrelated equations) and asked you to solve it for some variables? Something like this, where you are asked to solve for x and y:

$$\begin{aligned} 2x - y &= 1 \\ x + y &= 5 \end{aligned}$$

When you took algebra years ago, you probably learned different ways to solve these kinds of problems for x and y, such as substitution and elimination –

however now we are going to throw all of that out and teach you a *new* way so you'll never need to choose between different ways of solving these again – you'll have one way that works no matter what. Linear algebra takes the idea of solving these kinds of problems and allows us to find a general method that works no matter how many variables or equations we have. Better still, we can program computers with the method we develop so we can be extra lazy and not do anything by hand.

The first thing is to realize these algebraic equations express **linear combinations**, meaning they only include addition and scalar multiplication talked about in Section 2. There's no squaring or sin's or anything nonlinear. These linear combinations include addition (or subtraction) along with multiplication by coefficients. Notice that each equation is of the following form: $a_x x + a_y y = c$. In the first equation, $a_x = 2, a_y = -1$, and $c = 1$. In the second equation, $a_x = 1, a_y = 1$, and $c = 5$. Notice that each equation is kind of like an inner product: for each we take the variable x and multiply it by some scalar, then add it to the variable y multiplied by some scalar, and the result is equal to a single number.

Now for the tricky part: let's encode our algebraic variables x and y into a vector. The actual order doesn't matter, but we do need to be consistent once we decide on an order. Let's put x in the first component and y in the second: we use $\vec{b}$ to refer to the *vector* of algebraic variables: $\vec{b} = [x, y]^T$. Now, as an intermediate step, we can write each of the original equations as an inner product talked about in Section 8: the first equation can be written as

$$\langle [2, -1]^T, \vec{b} \rangle = \langle [2, -1]^T, [x, y]^T \rangle = 2 \cdot x - 1 \cdot y = 2x - y = 1$$

The second equation can be written as:

$$\langle [1, 1]^T, \vec{b} \rangle = \langle [1, 1]^T, [x, y]^T \rangle = 1 \cdot x + 1 \cdot y = x + y = 5$$

In general, each equation is of the form $\langle \vec{a}, \vec{b} \rangle = c$. What we will do next is try to "stack" each of these linear equations into a single expression that captures each of the individual equations. What changes among each equation is simply the coefficients vector $\vec{a}$ and the scalar c, however the general form (including the dimension) and the algebraic variables are the same for each expression. To write this as a single expression, we first create vectors for the coefficients:

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} x + \begin{bmatrix} -1 \\ 1 \end{bmatrix} y = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

We have encoded both equations into a single expression by writing the coefficients (and solutions) as vectors, with the algebraic variables as scalars (x and y are scalars because each refers to a single number). Now, we simply put our vector of algebraic variables on the right. So we have:

$$\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix} \implies \begin{aligned} 2x - y &= 1 \\ x + y &= 5 \end{aligned}$$

Pay very special attention to the way these operations are ordered here: we have defined a new operation, a **matrix-vector product**, which takes an inner product between a *row* of the matrix and the *column vector* on the right. We move *right, horizontally* along the matrix on the left and *down vertically* along the vector on the right, multiplying corresponding terms in each position and adding them all together. It may help to think of it in reverse: to get the 1st element in $\vec{c}$, which is on top, we start on the top-left element of A and the top element of $\vec{b}$: we multiply the corresponding items, and then move to the next, moving right along the matrix and down the vector. To get the 2nd element of $\vec{c}$, which is on the bottom, we start at the bottom-left of A and the top of $\vec{b}$. In general, to get the i-th element of a matrix-vector product, we take the inner product between the ith row of the matrix and the vector in question.

Written in terms of the individual elements:

$$\vec{c}_i = \sum_{j=1}^{n} A_{i,j}\vec{b}_j = \langle A_{i,:}, \vec{b} \rangle$$

Here, the notation ":" means "take all of the values in this position," which extracts the i-th row of the matrix A.

A matrix-vector product is an easy way to write a system of linear equations in a single line (remember we want to be *as lazy as possible*). We organize the information in a way where we put the algebraic variables in one vector, the coefficients in a matrix, and the solution in another vector. The general form of a system of linear equations is $A\vec{b} = \vec{c}$

$$\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix} \implies A\vec{b} = \vec{c}$$

Now the nice thing about this general system of linear equations is that $A\vec{b} = \vec{c}$ simply tells us the form of a system of linear equations – and we do not necessarily need to specify the dimensions or anything like this.

12 Why we care about matrix inverses

The general form $A\vec{b} = \vec{c}$ we wrote previously is very helpful, because it allows us to see exactly what we want to do. Remember that we want to find the values of x and y, which are loaded into the vector $\vec{b}$. If this was a *scalar* equation, we could solve for b by simply dividing both sides by a: $ab = c \implies b = c/a$. But this is a *vector* equation, so we need to be more careful: we can't really *divide* by a matrix the same way we divide by a scalar.

In fact, division isn't what we really care about at all: the reason we use division in the scalar case is that it "undoes" the transformation caused by multiplication. It just so happens that reversing the transformation induced by multiplication involves division. In general, whenever we have some equation of the form $f(x) = z$, we can view solving for x as taking the inverse:

$$f^{-1}(f(x)) = f^{-1}(z) \implies x = f^{-1}(z). \tag{4}$$

It may seem like we are introducing extra layers of complexity to something that doesn't need it, but trust me, this will allow us to be lazy in the future: this view of using division to perform an inverse function will allow us to extend the *idea* of division to multiple systems of equations at once. With this new idea of using functions to "undo" things, we can solve for **all variables at the same time**, enabling maximum laziness.

What we really want to do when we solve for $A\vec{b} = \vec{c}$ for $\vec{c}$ is *undo* what is caused by multiplication by A: if A is a single number, we "undo" multiplication by division. $a \cdot \frac{1}{a}x = x$. The number $\frac{1}{a}$ is the *multiplicative inverse*, meaning it "undoes" whatever multiplication by a does. So if $f(x) = ax$, then $f^{-1}(x) = \frac{x}{a}$. Consider $ax = b$, we apply f^{-1} to both sides, which gives $f^{(ax)} = f^{-1}(b) \implies \frac{ax}{a} = \frac{b}{a}$. The a's cancel, and we are left with $x = \frac{b}{a}$.

What we want to do is "undo" multiplication by the matrix A. We will write the mechanics of this later, but suppose such an inverse exists (we will notate it as A^{-1}). We want A^{-1} be defined so $A^{-1}A\vec{b} = \vec{b}$. Now we can solve for our algebraic variables by multiplying both sides by A^{-1}:

$$A\vec{b} = \vec{c} \implies A^{-1}A\vec{b} = A^{-1}\vec{c} \implies \vec{b} = A^{-1}\vec{c} \tag{5}$$

References

[1] Walter Rudin. *Principles of Mathematical Analysis*. 1953.

[2] Gilbert Strang and Linear Algebra. Linear algebra and its applications. *Academic Press, New York*, 14:181208, 1980.