ENGR 108: Introduction to Matrix Methods

Rohan Sikand

Autumn 2021

These are notes for Stanford's *ENGR 108: Introduction to Matrix Methods* (course website) taught by Stephen Boyd. The course covers the basics of vectors and matrices, solving linear equations, least-squares methods, and many applications. The textbook is [2] and and references made to the book is indeed the textbook. Important items are contained a gray box.

Contents

Section 1.

Introduction and Vectors

Lecture 1: `9/21/21`

We will go over some course logistics to start us off.

1.1 Course logistics

- Prerequisites: some calculus and basic programming. No need for prior linear algebra experience.
- Course requirements:
  - Weekly section
  - Weekly homework problem set with programming component in Julia.
  - Two lectures per week
  - Weekly "low-stakes" take-home quizzes. 48 hours to complete.
- The course textbook is *Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares*. We will cover chapter's 1-17.

1.2 Vector Notation

In this course, we treat a vector as follows:

> **Vector**: A vector is an ordered list of numbers (think array in computer code). The entries of a vector are called **components**. The number of elements in a vector is the **size** of the vector.

We use **n-vector** notation to describe a vector where n is the size of the vector. Say we have a vector that is of size 5. We call that vector a 5-vector.

Another point to note about vectors is that the size of the vector determines how many dimensions that vector is. A 5-vector is a five-dimensional vector. Spatially, you should think of each component of a vector as specifying the coordinate of the vector in that dimension (this is especially true for vectors in vector spaces since only vectors can exist there). For example, say we have the three dimensional vector $[-2, 3, 5]$. In three-dimensional Cartesian coordinate space, we can think of the -2 as denoting the location of the vector along the $x - axis$, the 3 along the $y - axis$, and the 5 along the $z - axis$.

There are also special types of vectors:

- unit-vector:
- zero-vector:
- one-vector:

1.3 Theoretical Underpinnings

Along with the usual notes from the class itself, I will occasionally introduce and describe a bit of the theoretical underpinnings we will learn in this class. Linear algebra contains a handful of formulas that, to the naked eye, makes it hard to decipher the true meaning and intuition behind the numbers. In addition, I will also lay out some concepts not covered in the course to assist in my understanding of the bigger picture of linear algebra[1].

To start us off on theoretical front, we will define a few concepts not used in this course but important for linear algebra in general. First, the definition of linear algebra itself:

> **Linear algebra**: Linear algebra is the study of *linear maps* on finite-dimensional *vector spaces* [1].

Seems a bit convoluted right? In English, this definition really is saying "Linear algebra is the study of mapping one collection of vectors to another".

> **Vector space**: A vector space is a set V which contains vectors in space where the properties of commutativity, associativity, additive identity, additive inverse, multiplicative identity, distributive properties hold.

That is, basically just a space where vectors live. This is useful because we can perform operations in a closed vacuum and in fact, operations on the space itself. Which leads us to what a **linear map**. We won't formally introduce a linear map (also referred to as a linear transformation) just yet as that is the topic discussed later, but briefly speaking, a linear map is simply a function that maps one vector space to another vector space while preserving the properties of additivity and homogeneity (which is what makes it linear).

[1] Linear algebra is the stepping stone from computational (high school) to abstract math (college). That is, in LA we can no longer get away with defining a bunch of computational steps for performing some operation for various cases. Instead, we define properties and theorems (as we will do in these notes) and use these to solve the problems. As such, you will need to understand the theoretical underpinnings (i.e. how this all works) well.

Section 2.

Linear Functions

We will talk about linear functions and an extension of a linear function called an affine function.

Note: it is important to understand that the word "function" is a generalization of a mapping from one set of things to another set of things. The type of those things is important in how a function is defined. A function that maps vectors to scalars (what we will do in this section) is called a scalar-valued function. A function that maps vectors to vectors (what we will do in section 7) is called a vector-valued function. You can have many different mappings that maps many different types of things so it is important to understand that the word "function" is just a general term.

2.1 Linear Functions

Of course, we use mathematical notation to declare the function itself and give its type (i.e. vector-valued, scalar-valued, etc.). Here is such notation[2]:

Function notation: We will use the following mathematical notation for a function in our case (i.e. with vectors):

$$f : \mathbf{R}^n \to \mathbf{R}$$

which means f is a function that maps real n-vectors to real scalars (scalar-valued function).

Wait what?! A function is something that maps a vector to a scalar? It may not make too much sense, but think of it like this. Let us use the continuous function notation we are all used to from high school. Say we have $f(3)$. This tells use the value of f at 3. This works similarly to the notation above in the case of vectors. Say we have a vector x. Then $f(x)$ represents the function f which takes in the vector x and returns the value of f at x (remember to think of a vector in cartesian space as the coordinates along each dimension; if it is easier, think of the function taking in the entries of the vector itself: $f(x) = f(x_1, x_2, \ldots, x_n)$). This value is in fact a scalar which is why we say a function maps a vector to scalars.

While we are all familiar with the $f(x)$ notation, we should start to become familiar with how we will describe functions in linear algebra dealing with vectors. Basically, **we need a way to specify what the value is for any possible input argument**.

- Traditionally (the familiar way), we can write a function like $f(x) = x_1 + x_2 - x_3$.
- We can also *describe* function based on attributes in order to guide how to find the value from its arguments.

[2]To give some more examples, $f : \mathbf{R}^n \to \mathbf{R}^n$ (vector-valued), etc.

- An example of this is the *sum function* which traditionally is written as $y = x_1 + \cdots + x_n$ (see how it is shorter just to describe the function as a sum function?). Moreover, it is also more generalizable. That is, we can utilize the description of a sum function for any n-vector instead of having to write out the entire formula.

So why so much detail for a seemingly frivolous concept regarding describing functions? Well, like the sum function, we will begin to familiarize ourself with important types of functions that play a key role in future topics.

Inner product function: Suppose a is an n-vector. We can define a scalar-valued function f of an n-vector by:

$$f(x) = a^T x = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

Essentially, the inner product function is simply a function which equates to the inner product between the vector itself and the coefficients of the function (which we can think of as a fixed vector a).

We now introduce an important concept and classification of a function.

Superposition and linearity: A function f satisfies the **superposition** if

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

holds for all numbers α, β and all n-vectors x, y. A function that satisfies superposition is called **linear**.

That is, a function is linear if you can take *any* two n-vectors and *any* numbers α, β, and show the superposition equality. This result (for a linear function) is also generalize-able to any number of vectors ($f(\alpha_1 x_1 + \cdots + \alpha_k x_k) = \alpha_1 f(x_1) + \cdots + \alpha_k f(x_k)$). Let us break this property down even further. We can think of superposition as a requirement for satisfying two properties relating to the argument $(\alpha x + \beta y)$. One involves the scalar-vector product and the other involves vector addition.

- **Homogeneity**: For any n-vector x and any scalar α, $f(\alpha x) = \alpha f(x)$.
- **Additivity**: For any n-vectors x and y, $f(x + y) = f(x) + f(y)$.

If the function can satisfy these two properties, it satisfies the superposition equality and is a linear function. Breaking the equality down into these two properties tells us the following. Homogeneity states that scaling the (vector) argument is the same as scaling the function value; additivity says that adding (vector) arguments is the same as adding the function values. Graphically, this helps us understand why superposition is the requirement for a function being linear since these both represent linear correlations.

You should think of linearity as a positive correlation between two entities. For example, say there is a linear relationship between x_1 and x_2. If you double x_1, then you should expect

the corresponding value in x_2 will also be doubled. An example of this would be $y = ax$ where $a \in \mathbb{R}$ (plug in numbers and see for yourself).

Another way of thinking about linearity in the context of linear transformations is described in 3Blue1Brown's video here.

2.2 Inner product representation of a linear function

There is a derivation (in the textbook) which shows that the inner product function (defined above as the inner product of its argument with some fixed vector) is linear. Extrapolating from this result, we can say that the converse is also true:

> If a function is linear, then it can be expressed as the inner product of its argument with some fixed vector.

That is, there exists some vector a (only one possible a exists) where $a^T x = f(x)$. We call $a^T x$, the **inner product representation** of f. See the derivation on page 31 of the textbook to see how this is true. In an applied sense, we can also use this result to determine linearity based on if we can find a vector a which satisfies $f(x) = a^T x$ for any n-vector x.

2.3 Affine functions

We now introduce another class of functions which is almost identical to linear functions with one small caveat.

Affine function: A linear function plus a constant. A function is affine if it can be expressed as $f(x) = a^T x + b$ for some n-vector a and scalar b (referred to as the offset).

Mathematically, a function is affine if it satisfies the superposition property but with one more requirement:

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

for all n-vectors x, y, and all scalars α, β **that satisfy** $\alpha + \beta = 1$. We can use this restricted superposition property for showing that functions are not affine (via $f(\alpha x + \beta y) \neq \alpha f(x) + \beta f(y)$ where $\alpha + \beta = 1$).

The difference between affine and linear functions is best showed in the following graphical representation.

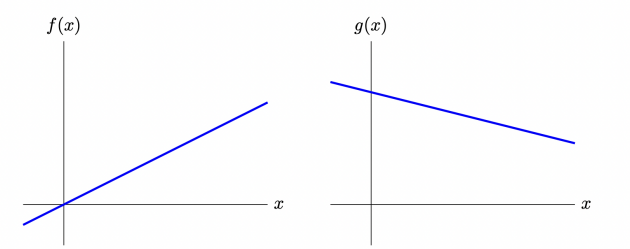

Figure 2.1 *Left.* The function f is linear. *Right.* The function g is affine, but not linear.

Figure 1: See how the y-intercept term is not 0 in $g(x)$? That is the offset term b.

Section 3.

Norm and distance

Lecture 2 and Lecture 3: `9/23/21`, `9/28/21`

This section introduces a handful of new concepts and corresponding formulae which are all defined below. See the textbook for derivations of each identity and example applications.

3.1 Norm

$$||x|| = \sqrt{x^T x}$$

3.2 Mean-squared value

$$\frac{x_1^2 + \cdots + x_n^2}{n} = \frac{\|x\|^2}{n}$$

Root Mean-Squared Value (RMS) is the same with a slight modification:

$$rms = \frac{\|x\|}{\sqrt{n}}$$

3.3 Standard deviation, de-mean, average

Standard deviation measures how much things vary from the mean.

$$\text{std}(x) = \textbf{rms}(\tilde{x}) = \frac{\left\|x - \left(\mathbf{1}^T x/n\right)\mathbf{1}\right\|}{\sqrt{n}}, \tilde{x} = x - \textbf{avg}(x)\mathbf{1}, \text{avg}(x) = \mathbf{1}^T x/n$$

3.4 Distance

$$\text{dist}(a, b) = \|a - b\|$$

3.5 Angle

$$\angle(a, b) = \arccos\left(\frac{a^T b}{\|a\|\|b\|}\right)$$

3.6 Correlation coefficient

$$\rho = \frac{\tilde{a}^T \tilde{b}}{\|\tilde{a}\|\|\tilde{b}\|}$$

Section 4.

Linear Independence

This is an important section and highlights most of the theoretical fundamentals of linear algebra topics that deals with vectors.

First a foremost, I recommend watching 3Blue1Brown's video on this topic (link here). His visual explanations shed light on the intuition behind the important but vague equations and concepts discussed here.

As we parse through the mathematical jargon, I will briefly summarize a few key points from 3Blue1Brown's video which should help with the intuition of what is to come.

Before we begin, we introduce a new way of thinking about vectors which should be helpful for understanding the concepts to come.

Think of vector coordinates as scaling basis vectors: we will introduce basis below but think of the unit vectors in $\mathbb{R}^2$ $((1,0),(0,1))$. Each vector in $\mathbb{R}^2$ can be expressed as a linear combination of the unit vectors. In other words, a vector in a vector space can be defined by how much it *scales* the unit vector.

Recall that we define a linear combination as follows:

Linear Combination: If $a_1, \ldots, a_m$ are n-vectors, and $\beta_1, \ldots, \beta_m$ are scalars, the n-vector

$$\beta_1 a_1 + \cdots + \beta_m a_m$$

is called a linear combination of the vectors $a_1, \ldots, a_n$. The scalars $\beta_1, \ldots, \beta_m$ are called the coefficients of the linear combination.

4.1 Basis

Basis can be be described in english as a generalization of the unit vectors in $\mathbb{R}^2$. In Linear Algebra, we deal with all sorts of spaces. In English, a basis is a list of vectors in a vector space that spans the vector space. By "spans" the vector space, we mean that the basis vectors via linear combinations can form any vector in the vector space.

Basis: A basis of a space V is a set of vectors $\{v_1, \ldots, v_n\}$ such that each vector $v \in V$ can be written uniquely as a linear combination of $v_1, \ldots, v_n$.

Parsing this through, we can see this is just a generalization and formalization of the unit vectors we are all familiar with in $\mathbb{R}^2$. How so? Recall that each vector in $\mathbb{R}^2$ can be expressed as a linear combination of the unit vectors $((1,0),(0,1))$. The remark at the beginning of this section (from 3Blue1Brown's video) really helps bring this concept to light.

4.2 Span

Now that we have both the mathematical definition and the intuition behind what a basis is, we can define a span. First, in english, a span of a vector space is **all of the possible vectors one can obtain from forming linear combinations with the basis vectors of that vector space**. Now in math:

Span: The set of all linear combinations of a list of vectors $v_1, \ldots, v_m$ (a vector space) in V is called the span of $v_1, \ldots, v_m$, denoted span$(\ v_1, \ldots, v_m)$. In other words,

$$\text{span}\,(v_1, \ldots, v_m) = \{a_1 v_1 + \cdots + a_m v_m : a_1, \ldots, a_m \in \mathbb{R}\}$$

4.3 Linear Independence

And we can now finally define linear independence:

Definition. (Linear Independence) A list $v_1, \ldots, v_m$ of vectors in V is called linearly independent if the only choice of $a_1, \ldots, a_m \in \mathbf{F}$ that makes $a_1 v_1 + \cdots + a_m v_m$ equal 0 is $a_1 = \cdots = a_m = 0$. If the list $v_1, \ldots, v_m$ of vectors is not linearly independent, it is considered linearly dependent.

In other words, a list of vectors is linearly independent if and only if all of them cannot be written as a linear combination of the others.

To understand this geometrically, let us think in $\mathbf{R^2}$. If we take all of the vectors in the list and plot them on the graph, the list is linearly independent if no two vectors sit on the same line. This makes sense because if they are on the same line, then one of them can be formed by *scaling* the other by forming a linear combination with nonzero scalars[3]. Again, 3Blue1Brown's video linked here illustrates this concept brilliantly and beautifully.

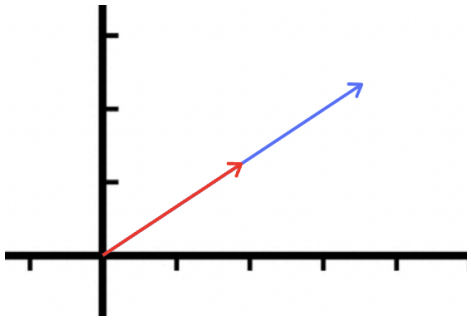

Figure 2: Example of two vectors that are linearly dependent. Clearly, one is just a scaled version of the other. Visually, they both sit on the same line (of the same slope of course).

[3] As 3Blue1Brown put it, one of the vectors in this case does not contribute anything to the span of the list

4.3.1 Independence-Dimension Inequality

From the definition of span and linear independence, we have the following corollary called the **independence-dimension inequality**: **no list of vectors that is larger than the size of the smallest basis (another list of vectors) can be linearly independent (Length of linearly independent list $\leq$ length of spanning list)**. If you parse this through, this makes sense since the definition of a basis is that it can be used in conjunction with linear combinations to form any vector in the vector space. Thus, any vector list that has more vectors than the basis must mean that some of them can be written as a linear combination of the other due to the definition of a basis.

4.3.2 Using matrices to determine linear independence

Here is an interesting thought and a teaser of what is to come:

> We can determine the scalars in a linear combination that show that a list of vectors is linearly independent or linearly dependent by representing it as a system of equations and solving it using matrices.

So this is a but ahead of what we learned thus far since we haven't introduced matrices. Say we have three vectors (v_1, v_2, v_3). We can test for linear dependence by forming the following linear combination:

$$x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + x_3\mathbf{v}_3 = \mathbf{0}$$

Now we need to solve for (x_1, x_2, x_3) and determine if at least one of them is nonzero. If so, then the list is linearly independent.

We have the homogeneous version of the famous system, $Ax = b$[4]:

$$A\mathbf{x} = \mathbf{0}$$

which we can use to represent the system of equations above by writing the vectors as columns in a matrix:

$$\begin{bmatrix} v_{1i} & v_{2i} & v_{3i} \\ v_{1j} & v_{2j} & v_{3j} \\ v_{1k} & v_{2k} & v_{3k} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

This is a matrix representation of a linear equation. We can solve this using various matrix operations which are discussed later. Solving linear equations via row reduction in matrices is discussed in a future section and is a critical part/application of linear algebra.

[4] which is really a vector-valued linear function represented using matrix form, as we will see later.

4.4 Orthonormal Vectors

4.5 Gram-Schmidt Process

We can determine if sets of vectors are linearly independent by running an algorithm called **Gram-Schmidt**.

4.6 Computational processes

This section included a lot of theoretical underpinnings that are essential to linear algebra. In an applied setting (such as during an exam), you might be given a set of vectors and will be asked to determine things relating to the topics discussed here (things such basis, linear independence, span, etc.). The definitions of these terms are in nature quite abstract and can be difficult to apply to problems. Fortunately, as we will see, there are several useful computational techniques/processes we can utilize to determine these things. However, these processes require an understanding of future topics such as matrices[5]. Thus, we will have sections later (such as sections 7 and 10) once we have learned those topics which discusses these computational processes.

[5]Indeed, a lot of what linear algebra is all about and what we will learn is about these processes relating to the concepts discussed here

Section 5.

Matrices

An important thing to realize is that we have been introducing and studying definitions that involve **sets** of vectors rather than just one vector specifically. Thus, we introduce here a compact way of representing a set of vectors which can then allow us to perform operations and determine things we wish to determine in a straightforward, computational manner. That is, we can represent a set of vectors using matrices. And recall that the definition of linear algebra is simply the study of linear transformations on finite lists of vectors. What this really means is that linear algebra is the study of operations (transformations) of matrices (lists of vectors). Actually in fact, as we will see in section 7, you can compactly represent a linear transformation with a matrix. As such, operations on such transformations is essentially matrix algebra which a lot of linear algebra courses focus on matrices.

5.1 Matrix-Vector Multiplication

We can represent many different things using matrix-vector multiplication (i.e. linear functions) so it is important to understand the fundamental operation.

If A is an $m \times n$ matrix and x is an n-vector, then the matrix-vector product $y = Ax$ is the m-vector y with elements [2]:

$$y_i = \sum_{k=1}^{n} A_{ik}x_k = A_{i1}x_1 + \cdots + A_{in}x_n, \quad i = 1, \ldots, m$$

This is best illustrated via an example [2]:

$$\begin{bmatrix} 0 & 2 & -1 \\ -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} (0)(2) + (2)(1) + (-1)(-1) \\ (-2)(2) + (1)(1) + (1)(-1) \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

Section 6.

Matrix Examples

6.1 Geometric Transformations

We mentioned that we can represent linear maps (in this case, vector-valued functions) using matrices of the form $y = Ax$. In applications, there are several useful transformations which we will cover here for the 2-dimensional case.

Remark. You should think of $Ax = b$ as an input-output *transformation*. That is, the matrix-vector product of Ax transforms the input vector x into another vector b. Thus, we can use matrices to represent linear transformations in a compact form.

Solving for one of these values (A, x, or b) is what linear algebra entails. In fact, in this class lots of problems have you determine what values of A transform x such that b is produced. And for that, I recommend you firmly understand how matrix-vector multiplication works. Really visualize what $Ax = b$ means: what values of A such that when A is multiplied with x produce the correct corresponding values in b?

6.2 Selectors

6.3 Incidence Matrix

6.4 Convolution

The convolution operation is a mathematical operation between two vectors that is, well, quite convoluted. There is a nice computational algorithm for calculating the convolution between two vectors by hand which we will get to.

Definition. (**Convolution**) The convolution of an n-vector a and an m-vector b is the $(n + m1)$-vector denoted $c = a * b$, with entries

$$c_k = \sum_{i+j=k+1} a_i b_j, \quad k = 1, \ldots, n + m - 1$$

where the subscript in the sum means that we should sum over all values of i and j in their index ranges $1, \ldots, n$ and $1, \ldots, m$, for which the sum $i + j$ is $k + 1$.

Seems a bit difficult to grasp at first which it is.

6.5 Flip-and-drag

Like mentioned, there is a nice computational algorithm for calculating the convolution between two vectors by hand which we will get to[6]. I briefly describe the algorithm here, but I highly recommend just looking at the visual explanation linked here.

[6] See here for a visual explanation

Say we have $c = a * b$. We can perform this calculation by reversing the components of one of either a or b. Then, take the vector that you reversed and line up the two vectors vertically such that the last component of the reversed vector lines up with the first component of the other vector. Here is an example where $a = (a_1, a_2)$ and $b = (b_1, b_2)$ where we choose to reverse b:

$$\begin{array}{cc} & a_1 \quad a_2 \\ b_2 & b_1 \end{array}$$

We now multiply the components that line up and sum each result. So in this case, we have $a_1 b_1$. This will be the first component of c. Now, here is where the "convolution" part comes in. We "drag" the reversed vector like a sliding window over to the right by one element:

$$\begin{array}{cc} a_1 & a_2 \\ b_2 & b_1 \end{array}$$

To get the second component, we have $a_1 b_2 + a_2 b_1$.This process repeats until the reversed vector falls off the edge.

$$\begin{array}{cc} a_1 & a_2 \\ & b_2 \end{array}$$

Thus, we have the 3-vector c:

$$c = (a_1 b_1, a_1 b_2 + a_2 b_1, a_2 b_2)$$

The convolution is an interesting operation and is widely used for signal processing in one-dimension and image processing in two-dimensions (yes, you can apply the convolution to a matrix[7]).

[7]Indeed, the Convolutional Neural Network is a pivotal algorithm that plays a huge role in computer visiona and deep learning

Section 7.

Linear Equations

In this section we consider vector-valued linear functions and introduce systems of linear equations.

In section 2, we learned about scalar-valued functions which maps vectors to real scalars. Here we introduce vector-valued functions:

Definition. (Vector-valued function) f is a function that maps real n-vectors to real m-vectors:

$$f : \mathbf{R}^n \to \mathbf{R}$$

We can also represent this like so: $f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$. It is important to note that each component in this function is itself a *scalar*-valued function.

An much already discussed example is the **matrix-vector product function**, $f(x) = Ax$.

The requirement that function satisfies superposition to make a function linear still applies. Now that we understand matrices, we actually have a proposition defined below in this section which states that if we represent a vector-valued function via some matrix A (i.e. $f(x) = Ax$), then the function is linear.

There are many examples in the real world that we can model via vector-valued linear functions[8]. See section 8.2 in the book for more.

7.1 Representing linear functions using matrices

Arguably the most fundamental equation in linear algebra is

$$Ax = b$$

Parsing this through, you can see that this is a method to represent a vector-valued linear function using a matrix A. To understand why, we must understand that a vector-valued function ($f : \mathbf{R}^n \to \mathbf{R}^n$) is a mapping between two sets of vectors. That is, given as input a vector, the function will *transform* it in someway and then output it. That is, a function represents a *transformation* of some sort. We can think of matrix-vector multiplication in a similar vein. This is because a matrix multiplied by a vector gives us another vector. So in that sense, we can construct the relevant values of A in $Ax = b$ to represent a certain vector-valued linear function (Yes!, we can represent an entire linear function via just the

[8]Note that a lot of exercises deal with finding the mapping A in $Ax = b$. Thus, you should understand the models discussed in section 8.2 of the book and be able to generalize

values in a matrix[9]). See 3Blue1Brown's video here for some intuition behind this idea. This is important because we can then perform a handful of matrix and vector operations to manipulate such mappings to obtain useful results of things we are interested in (i.e. solving a system of linear equations using matrices). Indeed, that is why we were are motivated to represent linear mappings via matrices because linear algebra is quite literally the study of *linear maps* on finite-dimensional *vector spaces* [1]. In fact, that is why matrices and their operations play a big role in linear algebra: because we can use them to represent things (linear mappings) we want to study further in concise way.

Because of all of this, we have the following proposition:

Proposition. If a function is linear, we can represent it using matrix-vector multiplication in the form $Ax = b$ ($f(x) = Ax$).

And from this we have the following corollary:

Corollary. If a function can be represented via a matrix A in the form $Ax = b$, then that function is linear.

This corollary makes a lot of sense. Say we have the linear function $f(x)$. Our input is x and the function is linear meaning the coefficients are multiplied by the input's x's. The coefficients are simply stored in the matrix A. From another perspective, it makes sense why we *cannot* store a nonlinear function in a coefficient matrix. This is because we may have something like x^2 in the function which means we cannot have a simple scalar multiplication consisting of the coefficient and the input to get the intended output value. Draw out an example to understand this.

7.2 Systems of linear equations

We now introduce the fundamental topic: **how to represent systems of linear equations via matrices**. We will later explore how to solve them using matrix operations in section 10.

Consider a set (also sometimes referred to as a "system" in application settings) of m linear equation in n unknowns: $\mu_1, \ldots, \mu_n$:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n &= b_2 \\ &\vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n &= b_m \end{aligned}$$

Above is your classic set of linear equations we've all seen from high school. We say that the numbers A_{ij} are the **coefficients** in the equations and the numbers b_i are called the

[9] And if you are wondering how to find those values, take one input and corresponding output and find the appropriate values by seeing how much each input is *scaled* to get to the output. Then, construct a matrix that is correctly indexed such that the product between itself and another vector gives the intended result based on the principles of matrix-vector multiplication.

right-hand-sides where, of course, the numbers x_i are **unknowns**. We can succinctly write these equations in matrix notation as

$$Ax = b$$

where A is an $m \times n$ matrix (coefficient matrix) and b is the m-representing the right-hand sides. We then aim to find an x that satisfies this equation. If indeed we do find an x where $Ax = b$ holds, we refer to x as a **solution** to the system of equations.

Importantly, a system of linear equations can either have:

- no solution
- one solution
- multiple solutions

which we will dive into shortly. First, let us see a concrete example.

Consider the following system of linear equations

$$x_1 + x_2 = 1, \quad x_1 = -1, \quad x_1 - x_2 = 0$$

We can write this via matrix notation written as $Ax = b$ as follows:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

Take another example:

The following equations

$$x_1 + x_2 = 1, \quad x_2 + x_3 = 2$$

can be written as $Ax = b$ like so:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Nicely, we can solve these equations using matrix operations as we will see later.

7.2.1 Over-determined and under-determined systems of linear equations

We can classify systems of linear equations as follows:

A system of linear equations is called:

- **over-determined** if $m > n$ (coefficient matrix A is tall).
  - This means that there are more equations that unknown variables.

- **under-determined** if $m < n$ (coefficient matrix A is wide).
  - This means that there are more unknown variables than equations.
- **square** if $m = n$ (coefficient matrix A is square).
  - This means that there are the same amount of unknown variables and equations.

Something to note is that a a system of equations with a zero right hand side, $Ax = 0$ is referred to as a **homogeneous** system of equations. Thus, $x = 0$ is a solution to any homogeneous system of equations. Interestingly enough, this can be tied back to determining linear independence discussed earlier. Specifically, take the following example of the coefficients of linear combinations.

Let $a_1, \ldots, a_n$ denote the columns of A. The system if linear equations $Ax = b$ can then be expressed as

$$x_1 a_1 + \cdots + x_n a_n = b$$

Parsing this through, you can see that b is a *linear combination* of $a_1, \ldots, a_n$ and coefficients $x_1, \ldots, x_n$. Solving for $Ax = b$ is the same as finding the coefficients to a linear combination that express b as a linear combination of $a_1, \ldots, a_n$. Thus, we can determine linear independence of $a_1, \ldots, a_n$ by seeing if there exists a nonzero vector b that satisfies $Ax = b$. Or, in other words, we can determine that $a_1, \ldots, a_n$ are linearly dependent if the only solution is $b = 0$ or $Ax = 0$ (homogeneous).

Remark. The connection between linear systems and linear independence makes sense since solving $Ax = b$ is really asking "can we write b is a linear combination of the columns of A?".

7.2.2 When does a system have 0, 1, or infinitely many solutions?

It is worth understanding the general cases of what types have systems have what solutions. To intuitively understand why, we must introduce some new terms[10].

Particularly, we have the following:

Let A be an $m \times n$ matrix. The system $Ax = b$ has:

- **No solution** if b does not belong to the column space $C(A)$ of A.
- Exactly **one solution** if $b \in C(A)$ and the null space $N(A)$ consists of only the zero vector.
- **Infinitely many solutions** if b belongs to $C(A)$ and the null space $N(A)$ has at least one nonzero vector.

We need to introduce a few new terms here to understand this.

[10] note that these new terms are not officially apart of ENGR 108 and are generally covered in more abstract linear algebra classes. Also, watch 3Blue1Brown's wonderful explanation of these concepts here.

Definition. (**Column Space**) The column space of an $m \times n$ matrix A is the *span* of the columns of A. It is denoted as $C(A)$ [3].

Definition. (**Null Space**) The null space of A, denoted $N(A)$, is the set of all solutions to the homogeneous system $Ax = 0$ [3].

To understand column spaces, recall the definition of span. Parsing the above definition, we say that a vector is in the column space if it can be written as a linear combination of the columns of A. From this logic, we have: for the linear system $Ax = b$, if b is in $C(A)$, then there exists some solution x. This makes sense since the Ax is just a matrix-vector multiplication resulting in a vector which is a linear combination of the columns (i.e. vectors) of A the scalars (i.e. coefficients) of x[11]. We do not consider the trivial (zero) solution to be a solution. Thus, we must understand null spaces. Intuitively, null spaces should be thought of in terms of linear transformations (since you can represent a linear transformation in a matrix like A). That is, the null space is simply the set of all vectors which, when transformed by A, result in the zero vector[12].

Now you can understand when a system has 0, 1, or infinitely many solutions. It is worth noting that if a system has > 1 solutions, it really has infinitely many solutions.

Interestingly enough, we also have the following conclusion:

Theorem. In <u>almost</u> all cases (not all though), overdetermined systems have no solution. In *all* cases, underdetermined systems have either no solution or infinitely many solutions (that is, if the underdetermined system has a solution, then it is has infinitely many solutions.

Thus, we can use the definitions of overdetermined and underdetermined systems to quickly come to conclusions about the consistency[13] of systems.

Also, note that the entire third part of ENGR 108 is dedicated to finding approximate solutions to overdetermined systems using a method called least squares.

[11] So the product Ax is a linear combination of the columns of A and the coefficients stored in x. Thus, if some x results in $Ax = b$, then we know that b is simply the linear combination described in the previous sentence. Thus, there exists at least one solution.

[12] If you think about it, this is like the analog of the y-intercept for linear maps.

[13] meaning how many solutions a system has.

Section 8.

Linear Dynamical Systems

Linear dynamical systems (LDS) is an application topic requiring matrix-vector multiplication. Essentially, we can represent sequences of states (e.g. time series) which are called dynamical systems.

Say we have a a sequence of n-vectors $x_1, x_2, \ldots$. We have something called the **state** which is a vector x_t at time t. x_{t-1} is the previous state and x_{t+1} is the next state. Examples might be economic output in n sectors, age distribution. These are dynamical systems. So what makes a dynamical system linear? A dynamical system is linear if the next state after the present one can be represented via matrix-vector multiplication (since Ax represents a linear transformation).

Formally, a dynamical system is linear if

$$x_{t+1} = A_t x_t, \quad t = 1, 2, \ldots$$

where A is the **dynamics matrix** of size $n \times n$.

Parsing this through, this is saying that, if we have a linear system, we can obtain the next vector in the sequence by *transforming* the current vector in some way. This transformation is compactfied in a matrix form, A. That is the mental model you should have:

Remark. the matrix A is used to represent some transformation that is applied to the current input x_t to obtain the next input, x_{t+1}.

8.1 Linear Dynamical Systems

Ok... now that we have some intuition described in words, let us formally define a linear dynamical system.

Definition. Linear Dynamical System

Suppose we have a sequence of n-vectors, $x_1, x_2, \ldots$. The index denotes the time t. The value at x_t is called the **state**. We call the sequence $x_1, x_2, \ldots$ the **state trajectory**. x_t represents the present state where x_{t+1} represents the next state and x_{t-1} represents the previous state.

A **linear dynamical system** is then a **model** for the sequence where each next state, x_{t+1} is given by the linear function:

$$x_{t+1} = A_t x_t$$

for $t = 1, 2, \ldots$. We call A_t the **dynamics matrix**.

Intuitively, we should think of x_t changing over time dynamically[14] (imagine a time series) where the equation given for x_{t+1} gives us a way to *update* the current state x_t to x_{t+1} as a function of x_t[15,16,17]. We call the system **time invariant** if A remains the same throughout all times t.

Why is this powerful? Because, all we need to know is the current state x_t and the matrix A_t and we can get our next state x_{t+1}. In other words: If we know the current value of x, we can find all future values [2].

Remark. So to construct a linear dynamical system, we need to construct a matrix A, the dynamics matrix, which represents how each state is transformed to produce the next state.

Additionally, we have:

Remark. $(A_t)_{ij}\,(x_t)_j$ is the contribution to $(x_{t+1})_i$ from $(x_t)_j$.

In words, this is saying that the jth component of the current state x_t, multiplied by the corresponding value in the matrix, $(A_t)_{ij}$, gives us part of what x_{t+1i} is equal to (since the matrix-vector multiplication is a summation of one row of the matrix and a column vector). This is best illustrated through examples such as the SIR model example in the book (section 9.3) [2].

8.2 Epidemic dynamics

Let us audit that example briefly. We have the following LDS modelled by:

$$x_{t+1} = \begin{bmatrix} 0.95 & 0.04 & 0 & 0 \\ 0.05 & 0.85 & 0 & 0 \\ 0 & 0.10 & 1 & 0 \\ 0 & 0.01 & 0 & 1 \end{bmatrix} x_t$$

where the x vectors are 4-vectors where each component represents the percentage of the population in one of four categories: susceptible (S), infected (I), recovered and immune (R), and deceased (D). Thus, each state is of the shape:

$$\boldsymbol{x_t} = \begin{bmatrix} S \\ I \\ R \\ D \end{bmatrix}$$

To get the next state x_{t+1}, we use matrix-vector multiplication:

[14] hence the name dynamical system.

[15] In that sense, the definition is recursive.

[16] Note that the function is linear which is why we call these linear dynamical systems. Recall from before that we discussed how linear transformations can be compactly represented in matrix form which is what we have here in Ax. See corollary 7.1.

[17] There is also *nonlinear systems* which are of interest because most real world systems are inherently non-linear in nature. There are whole fields of intriguing study dedicated to these systems such as chaos theory.

$$x_{t+1} = \begin{bmatrix} 0.95 & 0.04 & 0 & 0 \\ 0.05 & 0.85 & 0 & 0 \\ 0 & 0.10 & 1 & 0 \\ 0 & 0.01 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_t \\ I_t \\ R_t \\ D_t \end{bmatrix} = (0.95S_t + 0.04I_t + 0R_t, 0D_t, ...)$$

In other words, the first component of the next state x_{t+1} is the percentage of the population that is in the susceptible category (S) which is equal to 95% of the people who were susceptible in the current state (i.e. they are still susceptible) plus 4% of people who were infected in the current state (meaning they are recovered but not immune[18]. As such, if you parse through the indices, you can start to understand the remark above (remark 8.1).

[18] yes, the naming of the categories here is quite ambiguous.

Section 9.

Matrix Multiplication

We now introduce a fundamental operation: matrix multiplication.

9.1 Matrix-matrix multiplication

To multiply two matrices, A and B, their dimensions must be **compatible**. This means that the number of columns of A equals the number of rows of B. Say we have A which is $m \times p$ and B which is $p \times q$. We define the product matrix $C = AB$ of size $m \times n$ as follows:

Definition. (**Matrix Multiplication**)

$$C_{ij} = \sum_{k=1}^{p} A_{ik}B_{kj} = A_{i1}B_{1j} + \cdots + A_{ip}B_{pj}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

You can try to parse this through, but there several algorithms/rules which you should become familiar with.

Remark. To find the i, j element of C, go to the ith row of A and the jth column of B and take the dot product between these two vectors.

Here is an example:

$$\begin{bmatrix} -1.5 & 3 & 2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 0 & -2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3.5 & -4.5 \\ -1 & 1 \end{bmatrix}$$

To get the product, just follow the algorithm described in the remark above. For example, for $C_{1,2}$, we have $(-1.5)(-1) + (3)(-2) + (2)(0) = -4.5$.

9.2 Properties

The main one is asymmetry:

$$AB \neq BA$$

This does not even make sense unless A and B are of the same size. But even then, when you multiply it out, it may not equal.

We also have some others:

- Associativity: $(AB)C = A(BC)$
- Associativity with scalar multiplication: $\gamma(AB) = (\gamma A)B$

- Distributivity with addition: $A(B+C) = AB + AC$
- Transpose of product: $(AB)^T = B^T A^T$

Importantly, you should know that you can derive further properties from these main properties (e.g. $(A+B)(C+D) = AC + AD + BC + BD$)

Some others:

- $y^T(Ax) = \left(y^T A\right) x = \left(A^T y\right)^T x$

9.3 Matrix power

We can multiply a square matrix by itself to form AA or A^2. By convention, we also have $A^0 = I$. Some other properties include $A^k A^l = A^{k+l}$ and $\left(A^k\right)^l = A^{kl}$.

What happens if we have A^k where k is negative? We will discuss such a case later. This is called an inverse for the -1 case (which makes sense since $A^{-k} = \frac{1}{A^k}$).

Note that in this course, we restrict our study of matrix powers to integer powers. It is possible to have something like $A^{1/2}$ but that is out of scope for us.

9.4 Examples

9.4.1 Inner Products

In the beginning of the course, we introduced the inner product which is defined as $a^T b$. This is actually a special case of matrix multiplication as a^T is really just an $1 \times n$ matrix and b is a $n \times 1$ matrix. The result is a 1×1 matrix (i.e. scalar):

$$a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

which is really:

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

So that is where the definition of an inner product comes from. This makes sense since, in the remark above, we defined matrix multiplication as taking dot products between certain rows (the row vector a^T in this case) and certain columns (the row vector b in this case).

9.4.2 Linear dynamical systems

One cool thing about linear dynamical systems is that you only need one state input to determine the future states (since the future states depend on the calculation of the very next state).

For example, consider a time-invariant linear dynamical system:

$$x_{t+1} = Ax_t$$

We can find X_{t+2} like so:

$$x_{t+2} = Ax_{t+1}$$

But what didn't see before is that we can actually represent this via matrix powers and the original input:

$x_{t+2} = Ax_{t+1} = A\left(Ax_t\right) = A^2x_t$

To see how this is derived, recall that $x_{t+1} = Ax_t$. Thus, we just plug in for x_{t+1} in $x_{t+2} = Ax_{t+1}$ and get $A\left(Ax_t\right) = A^2x_t$. We can generalize this pattern as follows:

$$x_{t+\ell} = A^\ell x_t$$

where A^ℓ is the matrix that propagates the state forward ℓ time steps.

9.5 QR Factorization

This is a way to represent Gram-Schmidt in compact matrix notation.

Section 10.

Matrix Inverses

In this section, not only will we introduce a fundamental property of matrices (the inverse), but also we will see how to solve systems of equations with matrices.

10.1 Left and right inverses

Before we jump into matrix inverses, let us see it with scalars. We say that a number x that satisfies $xa = 1$ is the inverse of a. This what we are all used to from grade school. The inverse of a number is 1 over that number. This definition can be extended to matrices:

Definition. (**Left Inverse**) A matrix X that satisfies $XA = I$ is called a left inverse of A.

This has the following corollary:

Corollary. If A has a left inverse X then the columns of A are linearly independent.

We say that if a matrix has a left-inverse, it is left-invertible.

Example. If A has orthonormal columns (meaning A is orthogonal), then A^T is a left inverse of A since $A^T A = I$.

Proposition. (Left-invertibility and column independence) If A has a left inverse X then the columns of A are linearly independent. Or, vise versa: In order for X to exist, the left inverse of A, to exist, the columns of A are linearly independent (that is, a matrix has a left inverse if and only if its columns are linearly independent)[19].

Proof. Suppose $Ax = 0$ where the left inverse of A is C. Then, $0 = C(Ax) = (CA)x = Ix = x$. This makes sense since we are representing the linear combination of the vectors contained in the columns A with the scalars in the vector x. So if $x = 0$ in $Ax = 0$, then the only vector that satisfies $Ax = 0$ is the one with all 0s. This means that the only coefficients that make the linear combination of the column vectors equal to 0 are 0s (see 4.3.2). □

Something important to note:

Proposition. (Wide matrices are not left-invertible) A wide matrix $m < n$ is not left-invertible by the independence-dimension inequality (since a matrix has a left inverse if and only if its columns are linearly independent).

We can actually solve a system of linear equations by finding the left-inverse (there are many ways to do this as we will see but here is one of them).

Example. (Solving linear equations with a left inverse) Suppose $Ax = b$ where C is a left-inverse of A. We therefore know that $CA = I$. Thus,

$$Cb = C(Ax) = (CA)x = Ix = x.$$

[19] I do not understand this at all

Parsing this through, we try to multiply Cb. We can plug in for b with Ax and then do some matrix algebra to get $(CA)x$. From here, we know that $CA = I$ so we replace CA with I. Then, by the definition of the identity matrix, we get $Ix = x$. Therefore, the solution is $x = Cb$.

But how do we actually construct the inverse? We will see later.

Let us now introduce the right inverse:

Definition. (**Right Inverse**) A matrix X that satisfies $AX = I$ is called a right inverse of A.

A lot of the same examples/properties above hold for the right inverse as well.

Important ones:

Proposition. A matrix is right-invertible is its rows are linearly independent.

Thus:

Corollary. (Tall matrices are not right-invertible) A tall matrix $m > n$ is not right-invertible by the independence-dimension inequality (since a matrix has a right inverse if and only if its *rows* are linearly independent).

Let us now generalize this concept into one, unified concept of an *inverse*.

10.2 Inverse

Theorem. If a matrix is left-and-right-invertible, then the left and right inverses are unique and equal.

Proof. To see this, suppose $AX = I$ and $YA = I$. Then

$$X = (YA)X = Y(AX) = Y$$

□

When a matrix A has both a left inverse Y and a right inverse X, we call the matrix X/Y (note that $X = Y$) simply the inverse of A, and denote it as A^1. That is, in order for a matrix to be invertible, it must have a left and right inverse. These will be equal to each other and are also **unique** meaning that there only exists one A^{-1} if A is invertible.

Definition. (**Inverse**) The **inverse** of a matrix A, denoted as A^{-1} is defined such that

$$AA^{-1} = A^{-1}A = I$$

We refer to a square matrix that is invertible as a **nonsingular** matrix whereas a non invertible square matrix is referred to as a **singular** matrix.

Theorem. For a matrix A to be invertible, it must be square.

Proof. For A to be invertible, A must have a left inverse and a right inverse. Tall matrices are not right-invertible, while wide matrices are not left-invertible. Thus, A must be neither wide nor tall (so it must be square). □

By this, we also see that $\left(A^{-1}\right)^{-1} = A$.

We will explore this in more depth later in this section, but note that we can use the concept of an inverse to solve $Ax = b$.

Consider a square system of n linear equations with n variables: $Ax = b$. If A is invertible, then for any n-vector b, we can find x like so:

$$x = A^{-1}b$$

Note that this is the **unique** solution to $Ax = b$ (since A^{-1} is the right inverse of A[a]). It is unique since A^{-1} is the left inverse of A[b].

[a] Confused as to why this is the case. Where is the proof?
[b] Confused as to why this is the case. Where is the proof?

Proof.

$$\begin{aligned} \left(A^{-1}\right) Ax &= \left(A^{-1}\right) b \\ \left[\left(A^{-1}\right) A\right] x &= \left(A^{-1}\right) b \\ Ix &= \left(A^{-1}\right) b \\ x &= \left(A^{-1}\right) b \end{aligned}$$

□

10.2.1 Invertibility conditions

There are a lot of conditions regarding inverses of matrices. The nice thing about this is that if A is invertible then we know that:

- A is invertible
- The columns of A are linearly independent
- The rows of A are linearly independent
- A has a left inverse
- A has a right inverse

That is, if any of these hold, then all of them do.

10.2.2 Properties

Here are some properties of inverses:

- $(AB)^{-1} = B^{-1}A^{-1}$
- $\left(A^T\right)^{-1} = \left(A^{-1}\right)^T$
- $\left(A^{-1}\right)^k = A^{-k}$
- $A^0 = I$
  - Thus, $A^k A^l = A^{k+l}$

10.3 Computing the Inverse

There are several methods for computing the inverse (including a common method utilizing things we have not learned about at this point). The method introduced in the book is via QR factorization:

Computing the inverse via QR factorization Suppose A is invertible. This means that the columns of A are linearly independent. So we can perform QR factorization and get:

$$A = QR$$

We know that R is upper triangle and hence, invertible as well. We also know that since Q is orthogonal, we have $Q^TQ = I$. Therefore, we derive the following:

$$A^{-1} = (QR)^{-1} = R^{-1}Q^{-1} = R^{-1}Q^T$$

We will see how to compute R later in the least squares part of the course[20].

10.4 Solving systems of equations with matrices

We have now learned enough about matrices[21] to introduce arguably the most important use case of them: solving systems of equations. This concept is fundamental to linear algebra. Note that there are multiple ways to do this and we will explore a few.

10.4.1 Row reduction

The main approach is to use the row reduction algorithm.

[20] this is referring to the residual R seen in least squares right?

[21] Note that we may introduce a few new matrix operations along the fly if they are necessary.

Section 11.

Least Squares

We now introduce a powerful idea for *approximately* solving systems of linear equations that are over-determined. This idea is called **least squares** where we minimize the sum of squares of the errors in the equations. It has many applications since a lot of real world systems are overdetermined.

11.1 Least squares problem

Let us formalize the least squares problem.

Definition. Least Squares Problem

Suppose that we have an $m \times n$ matrix that is **tall** (meaning $m > n$, there are more rows and columns). This means that the system of linear equations is **over-determined** meaning that there are more equations (m) than variables to choose (n). Thus, these equations only have a solution if b is a linear combination of the columns of A. However, this does not work in most cases, as for most cases of b, there is no vector x that satisfies $Ax = b$.

Thus, we seek an **approximate solution** for x. We define the **residual vector**, r, as follows:

$$r = Ax - b$$

Our goal is now to find an x where r is as small as possible. Since r is a vector and not a scalar, to put things in terms of scalars, our actual goal is to choose an x that **minimizes the norm of the residual** ($||Ax - b||$). If successful, we have $Ax \approx b$.

Due to the definition of the norm, it is easier to define things terms of squares. Thus, we might as well minimize the norm of the residual squared (which is simply the sum of squares of the residual's components) since it yields the same result.

$$\|Ax - b\|^2 = \|r\|^2 = r_1^2 + \cdots + r_m^2$$

Thus, we can now formally introduce the **least squares** problem:

We want to find an n-vector $\hat{x}$ that minimizes $\|Ax - b\|^2$ (called the objective function, over all possible choices of x. This is denoted using the notation

$$\text{minimize} \quad \|Ax - b\|^2$$

where the *variable* is x (what we wish to find) and A, b are the *data*. Thus, $\hat{x}$ is a solution to the problem if

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2.$$

So in summary, our goal is to find a vector x such that the norm of the residual ($||Ax - b||$) is minimized[22].

You can also think of the matrix multiplication like so (this is called the column interpretation):

$$\|Ax - b\|^2 = \|(x_1 a_1 + \cdots + x_n a_n) - b\|^2$$

In this representation, the least squares problem is to find a linear combination of columns of A that is closest to b.

11.1.1 Geometric intuition behind least squares

We will cover linear least squares data fitting next section, but briefly here, we will also cover how least squares is used in the real world to gain some sort of intuition (watch this). Basically, say we have some data plotted on a cartesian plot. We want to find a line of best fit:

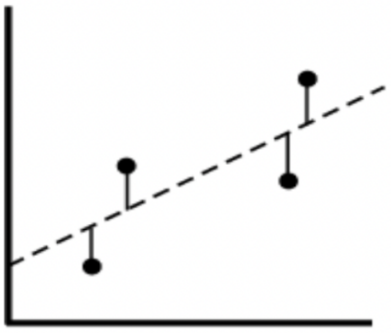

Figure 3: The lines extended from the function to each point are the residual errors. We sum those up, square them, and aim to minimize that value.

This is linear regression. But to solve linear regression (e.g. find the line of best fit), we can use linear least squares. We do this by constructing a linear equation of the form

$$y = ax + b$$

Note that in higher dimensions, we use the equation we are familiar with in this class: $y = A^T x + b$ but the principle still applies.

[22]Note that in data fitting contexts, this whole process is referred to as regression.

So how do we do this? Our goal is to construct the linear equation above by finding the vector x (in the 2d case, this is a 1-vector scalar called the slope) and bias vector b (y-intercept in the 2d case) that fit the data as best as possible. This is where least squares come into play (least squares determines how to construct the linear equation, e.g. line in two-dimensions).

Specifically, if we take the sum of differences at each point (that is the difference between the line and the data point and sum each one up), we want to minimize such a sum.

11.2 Solution

So how do we actually compute the solution? Recall the column interpretation of the least squares problem which said that the least squares problem is to find a linear combination of columns of A that is closest to b. This implies that for us to derive a closed-form solution to a least squares problem, we have one assumption:

Remark. Assumption in our closed-form solution to the least squares problem: The columns of A must be linearly independent[23].

11.3 Deriving the solution via calculus

So now let us derive the formula for the solution. Eventually, we will see that the formula is $A^{\dagger}b$ which will derive via calculus (it is a minimization problem after all).

From calculus, we know that any $\hat{x}$ must satisfy:

$$\frac{\partial f}{\partial x_i}(\hat{x}) = 0, \quad i = 1, \ldots, n$$

which we can express as:

$$\nabla f(\hat{x}) = 0$$

where $\nabla f(\hat{x})$ is the gradient of f evaluated at $\hat{x}$. This checks in with our understanding of basic optimization from calculus I: the derivative of the function at the minimum must be 0— we just generalize this here to higher dimensions.

We can express the general gradient in matrix form:

$$\nabla f(x) = 2A^T(Ax - b)$$

How did we derive this formula? Using calculus rules such as chain rules applied to matrices. Specifically, here is that derivation. We first write out the objective function as a sum:

$$f(x) = \|Ax - b\|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij}x_j - b_i \right)^2$$

[23] Wait so how do you compute the solution if the columns of A are linearly dependent?

which allows us to find $\nabla f(x)_k$ (i.e. the partial derivative of one x... now we just want to find the one where the gradient is 0). Taking this derivative (the partial derivative of f with respect to x_k.), we get:

$$\begin{aligned}\nabla f(x)_k &= \frac{\partial f}{\partial x_k}(x) \\ &= \sum_{i=1}^{m} 2\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)(A_{ik}) \\ &= \sum_{i=1}^{m} 2\left(A^T\right)_{ki}(Ax-b)_i \\ &= \left(2A^T(Ax-b)\right)_k\end{aligned}$$

which is exactly $2A^T(Ax-b)$. Thus, since we aim to find an x such that $\nabla f(x) = 0$, we have:

$$\nabla f(\hat{x}) = 2A^T(A\hat{x} - b) = 0$$

now here is the key part of the derivation: we can write this as:

$$A^T A\hat{x} = A^T b$$

Manipulating this using matrix operations to get $\hat{x}$ on the left-hand side, we have

$$\hat{x} = \left(A^T A\right)^{-1} A^T b$$

which is precisely

$$\hat{x} = \left(A^T A\right)^{-1} A^T b$$

which is equal to

Solution of a least squares problem:

$$\hat{x} = A^{\dagger} b.$$

And we are done. That is the solution (remember though, that this is still an approximation) derived via calculus on matrices. Recall that you can compute the value of $A^{\dagger}b$ via QR factorization. See the textbook for that.

Note that you can also derive this solution without calculus. See the textbook for that.

11.3.1 Example problem

We will briefly run through a small example. Consider the linear system $Ax = b$ where

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

This is a matrix representation of the over-determined set of linear equations:

$$2x_1 = 1, \quad -x_1 + x_2 = 0, \quad 2x_2 = -1$$

There is no solution since the system is over-determined. Thus, we can use least squares to find an approximate solution.

Recall that the least squares objective function can also be written using summation notation:

$$\|Ax - b\|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} x_j - b_i \right)^2$$

11.3.2 Computing $\hat{x}$ in practice

This subsection covers things not talked about in ENGR 108 but I think it is fruitful to have a discussion about such concepts when applying these methods in the real world.

We will see in the next chapter that we can use least squares to fit data in the real world (i.e. regression). This is relevant since many real world systems are over-determined.

In a first calculus class, you learn how to solve basic optimization problems by analytically finding where the derivative (i.e. the slope) is 0. This gets you the *exact* solution (closed-form solution). This type of method is analytical and referred to as a **direct method** in computational settings. However, one must realize that in the real world, it is rare that you will get a small enough system where you can go ahead and practically compute the exact solution using a direct method (picture a $100,000 \times 100,000$ matrix)[24]—see here. Furthermore, most systems in nature are nonlinear whose solutions must be computed iteratively (see here) instead of directly. That is why we see methods such as *gradient descent*.

So in theory, you can find the closed-form solution, $\hat{x}$, using the direct method:

$$\hat{x} = \left(A^T A\right)^{-1} A^T b = A^\dagger b$$

but in actuality, you will probably have to use an iterative solver such as gradient descent (which is why it is so prevalent in training models such as neural networks since they are large and nonlinear).

[24] Actually in fact, it is usually quicker to compute the solution via an iterative method at this scale.

As an aside, you may be asking "are we doing machine learning with least squares?". And the answer will be completely up to your preferences. There is no standard boundaries as to what is considered machine learning so it can be quite ambiguous. For example, one might think of machine learning as defining a model and then iteratively finding the minimum of the cost function across a training data distribution. But here we define the solution in closed-form... not using an iterative method.

Section 12.

Least Squares Data Fitting

Note on the goals of the course: the main application of this course and all of linear algebra in our context is to model some real world problem as a system of equations and solve it by recovering the unknown variable(s). We first begin our study by solving *linear* systems where the equations are linear. Then, we will solve nonlinear systems which is especially relevant since a lot of real world problems are inherently nonlinear. In general, there are two main methods for solving systems: *direct* and *iterative*. Direct methods yield a closed-form, exact solution and can usually be defined via formula–that is, it takes a finite amount of steps predefined based on the input variable(s). This would be all nice and dandy but a lot of real world problems cannot be solved via direct methods because of time complexity due to large inputs. Thus, we seek approximations of the exact solution using iterative methods (e.g. gradient descent). Iterative methods are also very relevant for solving nonlinear systems. So in summary, **our main goal in our study of linear algebra is to learn how to model a problem as a linear or nonlinear system and then learn how to solve for or approximate the solution to the system**.

Here we discuss a highly relevant application of least squares where we try to find a mathematical model given some observed data[25]. That is, we can apply the concepts of least squares to fit a plot of observed data (i.e. regression). Specifically, we can represent some sort of real world data as an overdetermined system of linear equations and compute the solution to the system using least squares.

12.1 The problem

The problem of using least squares for data fitting is quite intuitive. We will look at regression here but the core functionality can really be extended to meet the needs of other problems.

Say we have some data consisting of (x, y) pairs where the x's are the samples and the y's are the corresponding labels. We aim to find a function $f(x)$ such that $f(x) \approx y$ meaning that we aim to transform the input sample x, in such a way that it is close to y. Recall from section 7 that we can represent linear functions using a linear system of the form $Ax = b$. So that is what we do here: we represent the data fitting problem via $Ax = b$ where A contains the input samples, x contains the model's parameters, and b contains the corresponding labels. As such, we aim to solve for x, the model's parameters which, when multiplied by A, the inputs, will transform the inputs in such a way so that it is *close* to b, the labels. As such, we develop a cost function of the form

$$\text{minimize} \quad \|Ax - b\|^2$$

Note that it is common to use θ instead of x to represent the model's parameters:

[25] Yes, this is like machine learning. You might think of least squares as then the same thing as linear regression. The difference is that *linear* least squares is *one* of many ways to perform linear regression.

$$\text{minimize} \quad \|A\theta - b\|^2$$

Section 13.

Least Squares Classification

We now consider the use of least squares for fitting models who take in class values (i.e. classification).

13.1 The classification problem

For the binary classification problem, we want to approximate y by finding a function f:

$$f : \mathbf{R}^n \to \{-1, +1\}$$

where $y \approx f(x)$.

We have the following terms where y is the ground-truth value and $\hat{y}$ is the predicted value.

Definition. (**Prediction errors**)

- True positive. $y = +1$ and $\hat{y} = +1$.
- True negative. $y = -1$ and $\hat{y} = -1$.
- False positive. $y = -1$ and $\hat{y} = +1$.
- False negative. $y = +1$ and $\hat{y} = -1$.

13.1.1 Confusion matrix

We can form a **confusion matrix** to visualize how well a model fitted a given dataset by utilizing the counts of each prediction error defined above.

Say we have the dataset (generally the predictions on the test distribution are used for determining the values of a confusion matrix):

$$x^{(1)}, \ldots, x^{(N)}, \quad y^{(1)}, \ldots, y^{(N)}$$

and a model $\hat{f}$. For binary classification, we can assembly a 2×2 table where the columns correspond to the values predicted ($\hat{y}$) and the rows correspond to the ground-truth values (y). The entries of each slot in the table then give the count of each corresponding prediction error. Example:

	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	N_{tp}	N_{fn}	N_{p}
$y = -1$	N_{fp}	N_{tn}	N_{n}
All	$N_{\text{tp}} + N_{\text{fp}}$	$N_{\text{fn}} + N_{\text{tp}}$	N

where, for instance, N_{tp} corresponds to the count of true positives.

We can calculate useful performance metrics from the confusio matrix to gauge how well the model performed.

13.2 Least squares classifier

Section 14.

Multi-objective least squares

Here we introduce how to combine multiple least squares problems into one. That is, we consider the problem of choosing a vector that achieves a compromise in making two or more norm squared objectives small. The idea is widely used in data fitting, image reconstruction, control, and other applications [2]. For example, in a machine learning problem, we may wish to optimize more than one loss function (such as that the prediction is similar to the ground truth and that the the prediction is small).

To preface, the general idea here is to first combine the objectives into one least squares problem and solve it the "vanilla" way.

14.1 Multi-objective least squares problem

- Choose n vector x that minimizes k (multiple potentially) objectives. That is, we now have multiple objectives (goals) that we want to minimize.
- In single least squares, we wanted to do something like fit a line to some data. Here, we have *multiple* objectives:

$$J_1 = \|A_1 x - b_1\|^2, \ldots, J_k = \|A_k x - b_k\|^2$$

- To get this is ordinary least squares form, we formed a **weighted sum objective**:

$$J = \lambda_1 J_1 + \cdots + \lambda_k J_k = \lambda_1 \|A_1 x - b_1\|^2 + \cdots + \lambda_k \|A_k x - b_k\|^2$$

  where J is a scalar.

Section 1.

Resources

Here I will list some additional resources that can aid in the understanding of matrix methods and linear algebra in general.

- Kimberly Brehm's course on LA (link)
- MIT OCW 18.06
- Navigating Linear Algebra series
- 3B1B *Essence of Linear Algebra*
- Linear Algebra Done Right
- LA chapter in Deep Learning book

Additionally, see the Linear Algebra Toolkit which I like to describe as a calculator for linear algebra.

Some other things:

- Strang's textbooks on LA
- *The Art of Linear Algebra* graphic notes

Section 2.

Questions

Here are some lingering and fleeting questions that came to mind. Format-wise, each question is defined by the main bullet point and my thoughts/answer(s) are then jotted in sub bullet points.

- What is the difference between least squares data fitting and linear regression?
  - It seems that least squares gives us a *loss function* for the regression which needs to be performed via optimization. However, then what is the least squares solution then? Is that the exact perfect/loss minimized fit line? Why do we use optimization then in practice? Is the solution then not always calculable? See more here.

References

[1] Sheldon Jay Axler. *Linear algebra done right*. Vol. 2. Springer, 1997.

[2] Stephen Boyd and Lieven Vandenberghe. *Introduction to applied linear algebra: vectors, matrices, and least squares*. Cambridge university press, 2018.

[3] Stanford University Math Department. *Linear Algebra, Multivariable Calculus, and Modern Applications*. 2021.