COURSE NOTES (MTH 415, FALL 2025): APPLIED LINEAR ALGEBRA II

Mark Iwen

Chapters 1, 2, and Sections 3.1 – 3.7 are based on notes and figures initially scribed by Eric Brodsky while he was an undergraduate student working with me at Michigan State University in 2024.

M.A.I.

Contents

Chapter 1

Why We Should Care: Artificial Intelligence, and Data, Data, DATA!

Artificial Intelligence, which began being generally useful in the 2020's, resulted from the combination of three crucial historical developments: (*i*) the exponential increase in available computing power from the 1950's until the 2020's,[1] (*ii*) the development of machine learning techniques beginning in the second half of the 20th century (Neural Network methods in particular), and (*iii*) the collection of super-massive data sets for training and learning.[2] This book is meant to give the reader a solid introduction to the mathematics necessary to begin understanding developments (*ii*) and (*iii*) above. In particular, you will learn about the mathematics needed to understand what a neural network is and how the algorithms work that one might use to compile, process, analyze, and store the types of extremely super-massive datasets needed to train one well. Many of the mathematical topics needed are covered beginning in Chapter 2.

In this chapter we simply aim to prepare you to understand why that material is so important, as well as to state some application problems in a mathematical way that makes them easier to begin understanding more rigorously. Our main contention is this: learning the mathematics first makes all the application problems below much easier to learn about and begin solving later! However, we do understand that mathematics is difficult, and that it helps to have some solid motivation going into a long hike to help keep you trekking uphill until you reach the beautiful views nearer to the top of the mountain. We hope the following sections will help give you that motivation.

[1] Mainly due to steady innovations in integrated circuit manufacturing techniques over many decades – read up on Moore's law for a good time!

[2] Largely made possible by the development of modern communication infrastructure and the subsequent wide-scale adaptation of the internet beginning in the early 1990s.

1.1 Data, and What You Might Do with It

Let N be a positive integer. Herein we will let $[N]$ denote the first N non-negative integers from 0 to $N-1$, $[N] := \{0, \dots, N-1\}$ for any natural number $N \in \mathbb{N} = \{0, 1, 2, 3, \dots\}$. Our data herein will (almost always) be a vector of N numbers indexed by $[N]$. We will denote vectors with boldface letters. For example, $\mathbf{x} \in \mathbb{R}^N$ is a vector. We denote the entries of $\mathbf{x}$ by $x_j \in \mathbb{R}$. Pictorially, we have

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}.$$

In most settings we consider in this book vectors will be rich enough to represent the data we want to work with. This is primarily because, given the discrete and finite nature of digital computers, one can always simply vectorize other data one might have even if it isn't a vector to begin with. A related application example follows.

Example 1.1.1 (Image Classification Described with Vectors and Functions). *Suppose we want a model to separate pictures into two classes: pictures of cats and pictures of dogs. How can we describe this mathematically? Let's start with a picture of a cat. Assume this picture is* 1000 *pixels by* 1000 *pixels, and each pixel has some triple of color values associated to it (one for red, one for green, and one for blue), each a real number in the interval* $[0, 1]$*. Since a pixel is described by its three color values, each pixel in this image can be described a vector of length* 3*:*

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} \in \mathbb{R}^3$$

where r denotes the red value of the pixel, and so on. Doing this for each pixel in the image, we attain $1000 \times 1000 = 10^6$ *vectors of length* 3*. We can re-express this data as a single object by concatenating these vectors (in some arbitrary order, such as reading the pixel rows of the image left-to-right and top-to-bottom) into one large vector* $\mathbf{x}_{cat} \in \mathbb{R}^{3\times 10^6}$*. Hence, our cat picture is now simply a big vector.*

Now, let's focus on the question of classification. A classification model can be thought of as a function whose input is, e.g., a 1000×1000 *picture of a cat or a dog, and whose output is either "cat" or "dog". If we assign the label* 0 *to cats, and* 1 *to dogs (or the other way around, if you prefer cats!), then our classification question boils down to finding a function* $f : \mathbb{R}^{3\times 10^6} \to \{0, 1\}$ *such that, given a vectorized picture* $\mathbf{x}_{cat}$ *of a cat or a vectorized picture* $\mathbf{x}_{dog}$ *of a dog, we correctly get* $f(\mathbf{x}_{cat}) = 0$ *and* $f(\mathbf{x}_{dog}) = 1$*.*

We can use a similar framework for other sorts of problems. For example, the problem of reducing noise in a 1000×1000 cat picture can be viewed as a problem of finding a function $f : \mathbb{R}^{3\times10^6} \to \mathbb{R}^{3\times10^6}$ such that $f(\mathbf{x}_{\text{cat}})$ is "less noisy" than the original picture $\mathbf{x}_{\text{cat}}$.

There are a lot of specific image processing methods and techniques built around processing images as two-dimensional objects. For simplicity herein, however, we will use the flexibility of discrete representations to allow us to turn any image, etc., into a vector as an excuse to ignore non-vector data (i.e., we will vectorize everything). Though this can always be done, we note that it certainly shouldn't always be done... Nonetheless, it's generally useful enough that we will do it here. It also will make understanding the mathematics involved much easier, which we will take as an additional reason to assume that our datasets are almost always collections of vectors herein.

1.2 The Basics of Feed-forward Neural Networks (FNNs)

Continuing for the moment in the spirit of our first example above, we will now briefly take a detour to discuss what kinds of functions f one might actually build and evaluate with a computer to, e.g., classify images as in Example 1.1.1. FNNs provide exactly one such "computer friendly" class of functions that are also expressive enough to be able to do many useful tasks quite well. Given their value in artificial intelligence applications we will now take some time to explain what they are and how they depend on, and utilize, ideas from, e.g., both linear algebra and optimization. To begin we will first discuss the atomic building block of every neural network – the neuron.

1.2.1 Affine Functions and Single Neurons

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors in $\mathbb{R}^N$. We define the **inner product** of $\mathbf{x}$ and $\mathbf{y}$, denoted $\langle \mathbf{x}, \mathbf{y} \rangle$, to be the sum

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=0}^{N-1} x_j y_j \tag{1.1}$$

Definition 1.2.1 (Affine Functions)**.** *Fix* $\mathbf{w} \in \mathbb{R}^N$ *and a* $b \in \mathbb{R}$*. Then the* ***affine function*** *determined by* $\mathbf{w}$ *and* b *is the function* $a_{\mathbf{w},b} : \mathbb{R}^N \to \mathbb{R}$ *defined by*

$$a_{\mathbf{w},b}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{w} \rangle + b$$

Here $\mathbf{w}$ *is called the affine function's* **weight vector** *and* b *is called its* **bias***.*

Note that we can also write the above as a single inner product of two vectors in $\mathbb{R}^{N+1}$,

$$\langle \mathbf{x}, \mathbf{w} \rangle + b = \left\langle \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}, \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} \right\rangle .$$

We can also represent an affine function $a_{\mathbf{w},b} : \mathbb{R}^N \to \mathbb{R}$ graphically as in Figure 1.1.

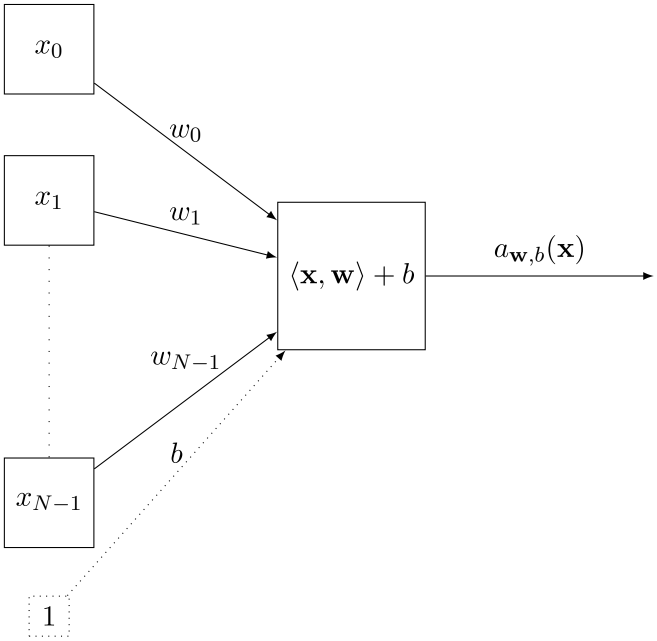
x_0
x_1
x_{N-1}
1
w_0
w_1
w_{N-1}
b
$\langle \mathbf{x}, \mathbf{w} \rangle + b$
$a_{\mathbf{w},b}(\mathbf{x})$

Figure 1.1: A graphical representation of an affine function $a_{\mathbf{w},b} : \mathbb{R}^N \to \mathbb{R}$. The first column of boxes represents the inputs (i.e., the entries of $\mathbf{x}$). The edge weights are the entry of the weight vector $\mathbf{w}$ that multiplies each corresponding input entry in the affine function's inner product (e.g., w_0 multiplies against x_0, etc.). The dotted box around the constant input 1 used here to include the bias b as an edge weight is often omitted.

Definition 1.2.2 (Neurons)**.** *A* ***neuron*** $\eta : \mathbb{R}^N \to \mathbb{R}$ *is a composition of an affine function* $a_{\mathbf{w},b}$ *with a nonlinear function* $\sigma : \mathbb{R} \to \mathbb{R}$ *given by*

$$\eta(\mathbf{x}) := \sigma(a_{\mathbf{w},b}(\mathbf{x})) = \sigma(\langle \mathbf{x}, \mathbf{w} \rangle + b)).$$

Note that a neuron is determined by two choices: the parameters $\mathbf{w} \in \mathbb{R}^N$ *and* $b \in \mathbb{R}$*, and the* **activation function** σ*.*

A neuron also admits the commonly used graphical representation in Figure 1.2. In Figure 1.2 the first column of boxes is called the **input layer** and the circle is called a **node** or **neuron**. Some typical choices of activation functions σ include the

- Perceptron (or Heaviside, or step function): $\sigma(y) = \begin{cases} 0 & \text{if } y \leq 0 \\ 1, & \text{if } y > 0 \end{cases}$
- Sigmoid: $\sigma(y) = 1/(1 + e^{-y})$
- Hyperbolic tangent: $\sigma(y) = \tanh(y)$

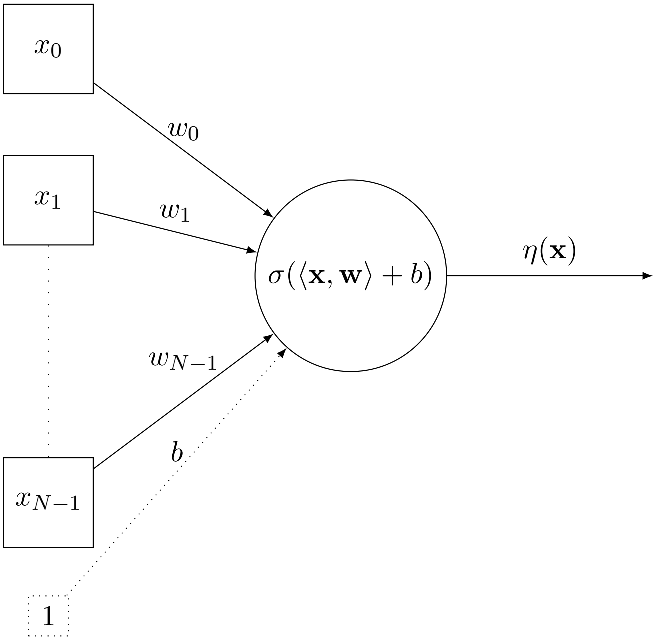

Figure 1.2: A graphical representation of a neuron. The first column of boxes represents the inputs (i.e., the entries of $\mathbf{x}$). The edge weights are the entry of the weight vector $\mathbf{w}$ that multiplies each corresponding input entry in the neuron's inner product (e.g., w_0 multiplies against x_0). Note in particular that a circle is used to represent a neuron here, as opposed to a box which is used to represent an affine function as per Figure 1.1. Again, the dotted box around the constant input 1 used here to include the bias b as an edge weight is often omitted.

- Rectified Linear Unit (ReLU): $\sigma(y) = \max(0, y)$
- Leaky ReLU: $\sigma_a(y) = \max(ay, y)$ with $0 \leq a < 1$.
- Absolute value (or modulus): $\sigma(y) = |y|$
- Smoothed versions of (leaky) ReLU to eliminate non-differentiability at $y = 0$.

Example 1.2.3 (A Simple Way to Smooth Non-Differentiability)**.** *Fix $a \in [0, 1)$ and $\alpha \in \mathbb{R}^+$, and define the function $g : \mathbb{R} \to \mathbb{R}$ to be*

$$g(y) = \begin{cases} a, & y < -\alpha \\ 1, & y > \alpha \\ a + \frac{1-a}{2\alpha} \cdot (y + \alpha), & -\alpha \leq y \leq \alpha \end{cases}$$

A smoothed leaky ReLU function, $\tilde{\sigma}_{a,\alpha}(x)$, can be defined to be

$$\tilde{\sigma}_{a,\alpha}(x) = \int_0^x g(y)\,dy. \tag{1.2}$$

Exercise 1.2.1. *The following problems concern the smoothed (leaky) ReLU function $\tilde{\sigma}_{a,\alpha} : \mathbb{R} \to \mathbb{R}$ defined in (1.2) with $a = 1/2$ and $\alpha = 1/4$.*

(a) Compute the integral in (1.2) and write down the resulting piecewise polynomial formula for $\tilde{\sigma}_{\frac{1}{2},\frac{1}{4}}(x)$. What is $\tilde{\sigma}_{\frac{1}{2},\frac{1}{4}}(1)$?

(b) Plot $\tilde{\sigma}_{\frac{1}{2},\frac{1}{4}}$ together with the leaky ReLU function $\sigma_{\frac{1}{2}}$.

Given an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ we will extend it to a function $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ for any given $N \in \mathbb{N}$ entrywise by

$$\sigma(\mathbf{x}) := \begin{pmatrix} \sigma(x_0) \\ \sigma(x_1) \\ \vdots \\ \sigma(x_{N-1}) \end{pmatrix}.$$

We will now continue to build on this notation in order to help combine multiple neurons into more complicated (and useful!) functions.

1.2.2 Layers of Neurons, and Some Helpful Matrix Notation

A **matrix** $W \in \mathbb{R}^{N \times d}$ is a table of data with N rows and d columns. We denote the entry in the j^{th} row and k^{th} column of W by $W_{j,k} \in \mathbb{R}$ for all $j \in [N]$ and $k \in [d]$. We denote the j^{th} row of W, which is a vector in $\mathbb{R}^d$, by $W_{j,:} \in \mathbb{R}^d$. Similarly, we denote the j^{th} column of W, which is a vector in $\mathbb{R}^N$, by $W_{:,j} \in \mathbb{R}^N$. We can also build a matrix out of vectors. Given d vectors $\mathbf{w}_0, \ldots, \mathbf{w}_{d-1} \in \mathbb{R}^N$, we can write the $N \times d$ matrix whose j^{th} column is $W_{:,j} = \mathbf{w}_j$ for all $j \in [d]$ as

$$\begin{pmatrix} | & & | \\ \mathbf{w}_0 & \cdots & \mathbf{w}_{d-1} \\ | & & | \end{pmatrix} \in \mathbb{R}^{N \times d}.$$

Given a matrix $W \in \mathbb{R}^{N \times d}$ and a vector $\mathbf{y} \in \mathbb{R}^N$ we will also denote by $(W|\mathbf{y}) \in \mathbb{R}^{N \times (d+1)}$ matrix whose first d columns are the columns of W, and whose $(d+1)^{\text{st}}$ column is $\mathbf{y}$.

The **transpose** of a matrix $W \in \mathbb{R}^{N \times d}$, denoted by $W^T \in \mathbb{R}^{d \times N}$, is the $d \times N$ matrix with entries given in terms of W by $(W^T)_{j,k} = W_{k,j}$ for all $j \in [d]$ and $k \in [N]$. That is, we swap the roles of rows and columns so that, e.g., $W_{j,:} = W^T_{:,j}$ for all $j \in [N]$.

Finally, a matrix $W \in \mathbb{R}^{N\times d}$ also always represents a linear function $W : \mathbb{R}^d \to \mathbb{R}^N$ where $W(\mathbf{x}) = W\mathbf{x} \in \mathbb{R}^N$ has entries given by

$$(W\mathbf{x})_j := \sum_{k\in[d]} W_{j,k}x_k$$

for all $j \in [N]$.

Exercise 1.2.2. *Let $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^d$, and $W \in \mathbb{R}^{N\times d}$. Show that $\langle \mathbf{x}, W\mathbf{y}\rangle = \langle W^T\mathbf{x}, \mathbf{y}\rangle$.*

We can also represent a matrix graphically as multiple affine functions. Let $W \in \mathbb{R}^{N\times d}$ and $\mathbf{x} \in \mathbb{R}^d$. Then we can express the matrix-vector product $W\mathbf{x} \in \mathbb{R}^N$ with the diagram in Figure 1.3

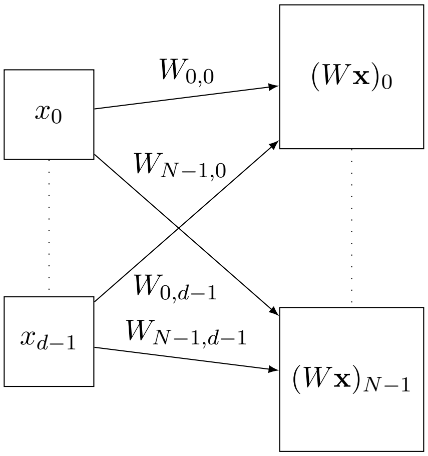

Figure 1.3: A graphical representation of a matrix $W \in \mathbb{R}^{N\times d}$ as an input layer of width d connected directly to a linear output layer of width N.

We now have enough notation to define and represent a single layer of neurons.

Definition 1.2.4 (A Layer of Neurons)**.** *A **layer of neurons** $\ell : \mathbb{R}^N \to \mathbb{R}^d$ is determined by a collection of d weight vectors $\mathbf{w}_0, \dots, \mathbf{w}_{d-1} \in \mathbb{R}^N$, d biases $b_0, \dots, b_{d-1}$, and a choice of activation function $\sigma : \mathbb{R} \to \mathbb{R}$. We call d the **width** of ℓ. The layer ℓ is defined using these parameters by*

$$\ell(\mathbf{x}) := \begin{pmatrix} \sigma(\langle \mathbf{x}, \mathbf{w}_0\rangle + b_0) \\ \vdots \\ \sigma(\langle \mathbf{x}, \mathbf{w}_{d-1}\rangle + b_{d-1}) \end{pmatrix} = \sigma \begin{pmatrix} \langle \mathbf{x}, \mathbf{w}_0\rangle + b_0 \\ \vdots \\ \langle \mathbf{x}, \mathbf{w}_{d-1}\rangle + b_{d-1} \end{pmatrix} = \sigma(W\mathbf{x} + \mathbf{b}),$$

where $W^T = \begin{pmatrix} | & & | \\ \mathbf{w}_0 & \cdots & \mathbf{w}_{d-1} \\ | & & | \end{pmatrix} \in \mathbb{R}^{N\times d}$ *and* $\mathbf{b} = \begin{pmatrix} b_0 \\ \vdots \\ b_{d-1} \end{pmatrix}$. *Note that* $\ell : \mathbb{R}^N \to \mathbb{R}^d$ *is effectively created by stacking* d *different neurons* $\eta_0, \dots, \eta_{d-1} : \mathbb{R}^N \to \mathbb{R}$ *into a vector. Here* $W \in \mathbb{R}^{d\times N}$ *is called the layer's* **weight matrix** *and* $\mathbf{b}$ *is called the layer's* **bias vector**.

Above can also write $\ell(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$ as $\ell(\mathbf{x}) = \sigma\left(\tilde{A}\begin{pmatrix}\mathbf{x} \\ 1\end{pmatrix}\right)$, where $\tilde{A} = (W|\mathbf{b}) \in \mathbb{R}^{d\times(N+1)}$. Thus, if we define the affine function $A : \mathbb{R}^N \to \mathbb{R}^d$ by $A(\mathbf{x}) \coloneqq \tilde{A}\begin{pmatrix}\mathbf{x} \\ 1\end{pmatrix} = W\mathbf{x}+\mathbf{b}$, we may further write ℓ compactly as a composition of σ and A, i.e., $\ell(\mathbf{x}) = \sigma(A(\mathbf{x})) = (\sigma \circ A)(\mathbf{x})$. This compositional form will be used below. Finally, we note that one can also represent a layer of d neurons graphically as per Figure 1.4.

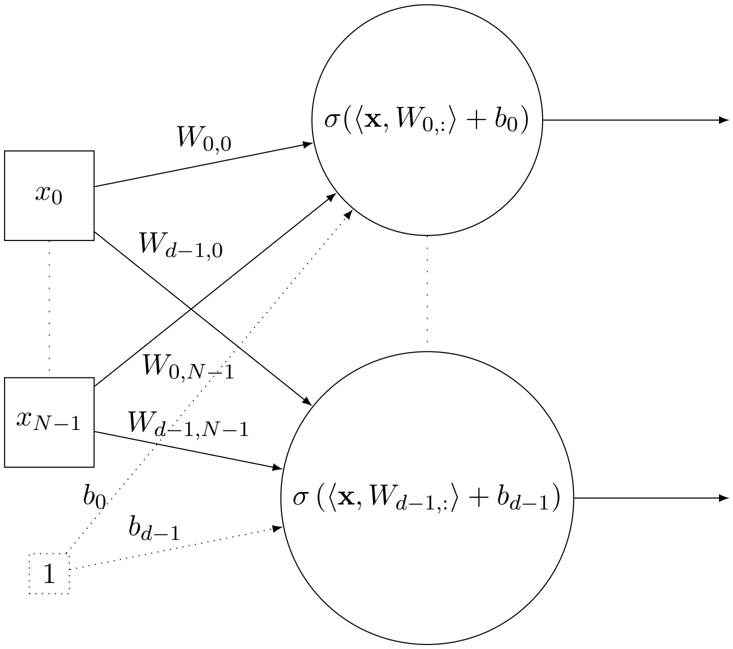
$\sigma(\langle \mathbf{x}, W_{0,:}\rangle + b_0)$
$W_{0,0}$
x_0
$W_{d-1,0}$
$W_{0,N-1}$
x_{N-1}
$W_{d-1,N-1}$
b_0
$\sigma\left(\langle \mathbf{x}, W_{d-1,:}\rangle + b_{d-1}\right)$
b_{d-1}
1

Figure 1.4: A graphical representation of a layer of neurons $\ell : \mathbb{R}^N \to \mathbb{R}^d$ defined by $\ell(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$ with weight matrix $W \in \mathbb{R}^{d\times N}$ and bias vector $\mathbf{b} \in \mathbb{R}^d$. Here the input layer of width N connects to a layer of d neurons.

1.2.3 Feed-Forward Neural Networks (FNNs) in Full Generality

Informally, a FNN is a series of layers of neurons with each layer feeding its outputs "forward" into the layer following it. From the discussion of neuron layers above, we can therefore choose to describe an FNN as a concatenation of functions that alternate between affine functions and the activation function. More formally, one can define FNNs as follows.

Definition 1.2.5 (Feed-forward Neural Network (FNN))**.** *A **Feed-forward Neural Network (FNN)** $f : \mathbb{R}^N \to \mathbb{R}^{d_L}$ is determined by an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, a **depth** $L \in \mathbb{N}$, and layer widths $d_0, \ldots, d_L$. It contains an input layer with N inputs, L layers of neurons (often called **hidden layers**), and a final linear output layer with d_L outputs. More specifically, let $\ell^0 : \mathbb{R}^N \to \mathbb{R}^{d_0}$ and $\ell^j : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j}$ $\forall j \in \{1, \ldots, L-1\}$ be L layers of neurons, and let $A^L : \mathbb{R}^{d_{L-1}} \to \mathbb{R}^{d_L}$ be an affine function defined by $A^L(\mathbf{y}) := W^L\mathbf{y} + \mathbf{b}^L$ for $W^L \in \mathbb{R}^{d_L \times d_{L-1}}$, $\mathbf{b}^L \in \mathbb{R}^{d_L}$. The resulting FNN of depth L, f, is then given for all $\mathbf{x} \in \mathbb{R}^N$ by*

$$\begin{aligned} f(\mathbf{x}) &= \left(A^L \circ \ell^{L-1} \circ \ell^{L-2} \circ \cdots \circ \ell^1 \circ \ell^0\right)(\mathbf{x}) \\ &= \left(A^L \circ \sigma \circ A^{L-1} \circ \sigma \circ A^{L-2} \circ \cdots \circ \sigma \circ A^0\right)(\mathbf{x}), \end{aligned}$$

where $A^{L-k}(\mathbf{y}) = W^{L-k}(\mathbf{y}) + \mathbf{b}^{L-k}$, with $W^{L-k} \in \mathbb{R}^{d_{L-k} \times d_{L-k-1}}$ and $\mathbf{b}^{L-k} \in \mathbb{R}^{d_{L-k}}$ for all $k \in [L]$, and $A^0(\mathbf{y}) = W^0\mathbf{y} + \mathbf{b}^0$, with $W^0 \in \mathbb{R}^{d_0 \times N}$ and $\mathbf{b}^0 \in \mathbb{R}^{d_0}$.

Even after fixing the activation function σ we note that FNNs are functions that depend on a potentially huge number of parameters. Using our notation from above, the number of parameters in a FNN f is equal to the sum of the number of weights in the matrices W^j and the number of biases in the vectors $\mathbf{b}^j$ for all $j \in [L+1]$. Recall that when $j > 0$, $W^j \in \mathbb{R}^{d_j \times d_{j-1}}$ and $\mathbf{b}^j \in \mathbb{R}^{d_j}$, and when $j = 0$, $W^0 \in \mathbb{R}^{d_0 \times N}$ and $\mathbf{b}^0 \in \mathbb{R}^{d_0}$. Thus, the total number of parameters for a depth L FNN f with input layer width N and hidden layer widths $d_0, d_1, \ldots, d_L$ is

$$\#\text{ FNN parameters} = d_0(N+1) + \sum_{j=1}^{L} d_j(d_{j-1}+1).$$

Finding a good way of choosing all of these parameters during training so that the resulting trained FNN is capable of, e.g., correctly classifying cat versus dog pictures is usually accomplished via optimization techniques. Techniques one can use to help reduce the number of these parameters in order to save space when storing a previously-trained FNN is something we will discuss more in, e.g., Sections 2.5 and 3.1. We urge you to keep reading to learn about these useful tricks, and more!

For now though, we will simply try to mitigate the fact that the general definition of a depth L FNN given above is rather complicated. In order to help digest it, let's consider some examples. Our first example will be that of a **shallow** FNN (that is, of an FNN of depth $L = 1$).

Example 1.2.6 (A Shallow FNN $f : \mathbb{R} \to \mathbb{R}$)**.** *A shallow (i.e., $L = 1$) FNN $f : \mathbb{R} \to \mathbb{R}$ will have the form*

$$f(x) = b^1 + \sum_{j=0}^{d_0-1} w_j^1 \sigma\left(w_j^0 x + b_j^0\right). \tag{1.3}$$

where $b^1 \in \mathbb{R}$ is the single output layer bias (the output width is $d_1 = 1$), b^0_j for all $j \in [d_0]$ are the biases of the single layer of neurons of width d_0, and where the weights of the layer of neurons and the output layer are $w^0_j, w^1_j \in \mathbb{R}$ for all $j \in [d_0]$, respectively.

Example 1.2.7 (The Graphical Representation of a Shallow FNN $f : \mathbb{R}^3 \to \mathbb{R}^2$)**.** *For a graphical representation of a shallow (i.e., depth $L = 1$) FNN $f : \mathbb{R}^3 \to \mathbb{R}^2$ with widths $d_0 = 2$ and $d_1 = 2$ see Figure 1.5. Note that such a network will be determined by two weight matrices $W^0 \in \mathbb{R}^{2\times 3}$, $W^1 \in \mathbb{R}^{2\times 2}$ and two bias vectors $\mathbf{b}^0, \mathbf{b}^1 \in \mathbb{R}^2$. Hence, it has a total of* 14 *parameters.*

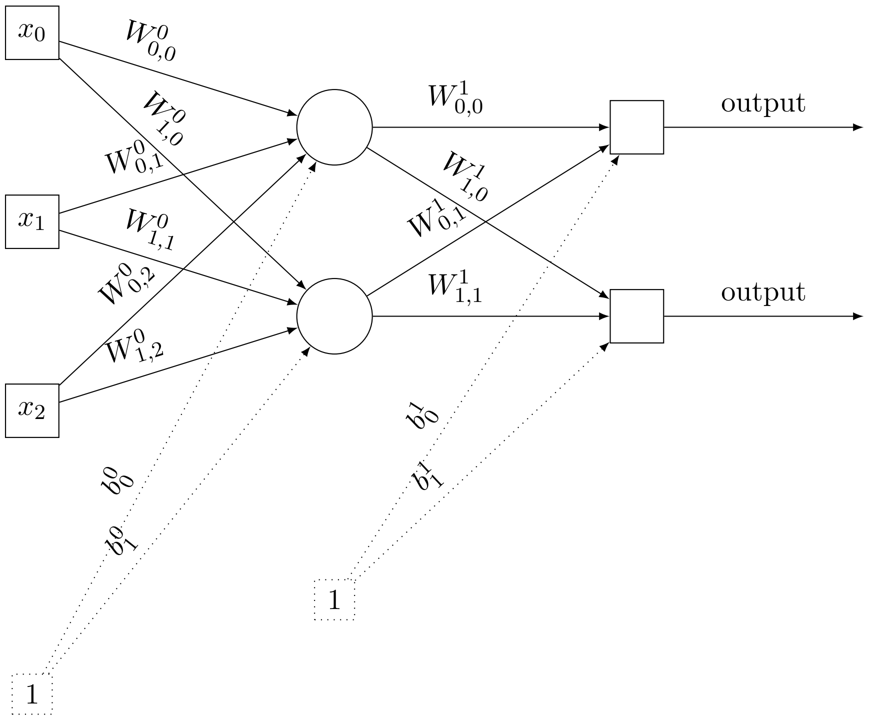

Figure 1.5: An example of a shallow neural network $f : \mathbb{R}^3 \to \mathbb{R}^2$. We call a depth 1 FNN **shallow**. The leftmost layer is the input layer with $N = 3$ inputs. The middle layer, which is the only nonlinear layer in this diagram, is a hidden layer of neurons with $d_0 = 2$ neurons. The right layer is the output layer with $d_L = 2$ outputs.

Example 1.2.8 (The Graphical Representation of a Depth $L = 2$ FNN $f : \mathbb{R}^2 \to \mathbb{R}^2$)**.** *For a graphical representation of a depth $L = 2$) FNN $f : \mathbb{R}^2 \to \mathbb{R}^2$ with widths $d_0 = 3$, $d_1 = 2$, and $d_2 = 2$ see Figure 1.6. Such a network will be determined by three weight matrices $W^0 \in \mathbb{R}^{3\times 2}$, $W^1 \in \mathbb{R}^{2\times 3}$, $W^2 \in \mathbb{R}^{2\times 2}$ and three bias vectors $\mathbf{b}^0 \in \mathbb{R}^3, \mathbf{b}^1, \mathbf{b}^2 \in \mathbb{R}^2$. Hence, it has a total of* 23 *parameters.*

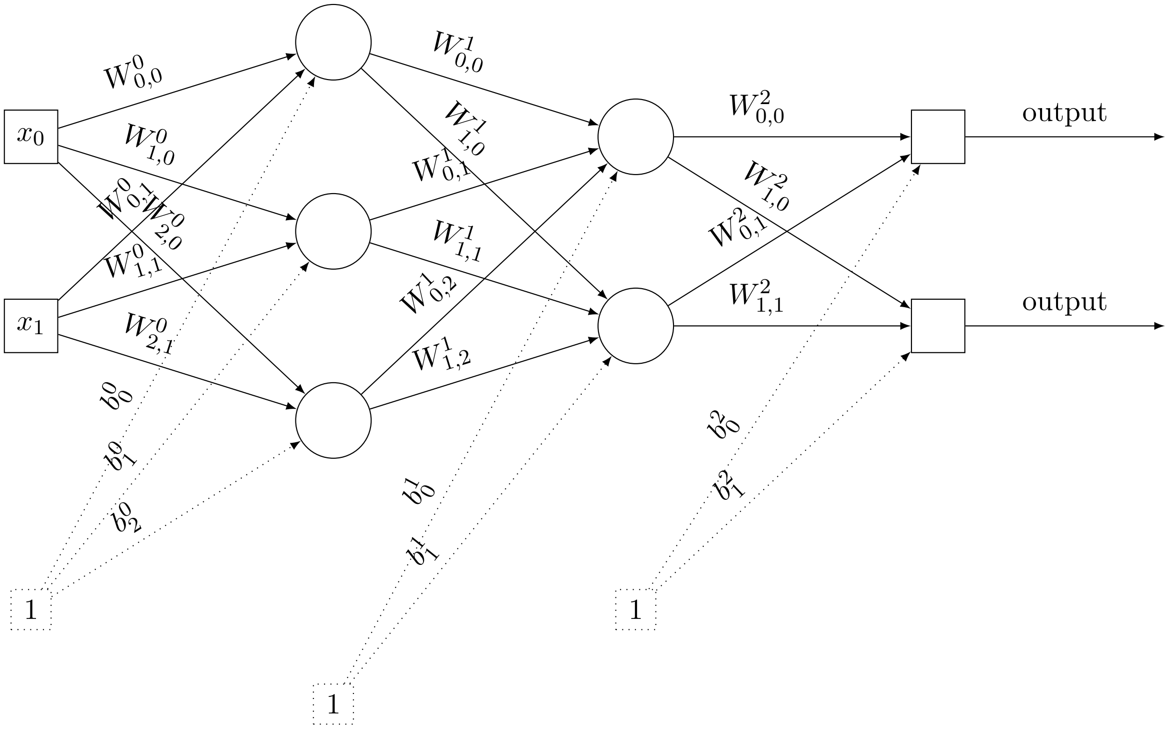

Figure 1.6: An example of a neural network $f : \mathbb{R}^2 \to \mathbb{R}^2$ of depth $L = 2$. The leftmost layer is the input layer with $N = 2$ inputs. The second layer from the left is the first hidden layer of neurons, which has $d_0 = 3$ neurons. The third layer from the left is the second hidden layer of neurons, which has width $d_1 = 2$, and the rightmost layer is the linear output layer, which has $d_L = d_2 = 2$ outputs.

Exercise 1.2.3. *Draw the graphical representation of a shallow neural network $f : \mathbb{R} \to \mathbb{R}$ of width $d_0 = 5$. How many parameters does it have?*

Exercise 1.2.4. *Draw the graphical representation of a depth $L = 3$ neural network $f : \mathbb{R} \to \mathbb{R}$ with widths $d_0 = 2, d_1 = 2, d_2 = 2$. How many parameters does it have?*

We will now briefly discuss why choosing, e.g., a greater value for its depth L might allow a FNN to "work better" at a variety of tasks. This is directly linked to the notion of the "expressivity" of an FNN.

Some Basics Concerning the Expressivity of FNNs

In practice the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is always chosen to be a nonlinear function. The reason why is directly linked to the notion of the "expressivity" of an FNN. Suppose for example that we choose $\sigma : \mathbb{R} \to \mathbb{R}$ in (1.3) to be linear so that $\sigma(y) = ay + c$ for some $a, c \in \mathbb{R}$. Substituting this activation function into (1.3) we obtain

$$\begin{aligned} f(x) &= b^1 + \sum_{j=0}^{d_0-1} w_j^1 \sigma\left(w_j^0 x + b_j^0\right) = b^1 + \sum_{j=0}^{d_0-1} w_j^1 \left[a\left(w_j^0 x + b_j^0\right) + c\right] \\ &= \underbrace{\left(\sum_{j=0}^{d_0-1} w_j^1 a w_j^0\right)}_{=:\ \tilde{a}} x + \underbrace{\left(b^1 + \sum_{j=0}^{d_0-1} w_j^1 (a b_j^0 + c)\right)}_{=:\ \tilde{c}} = \tilde{a}x + \tilde{c}, \end{aligned}$$

with the two new constants $\tilde{a}, \tilde{c} \in \mathbb{R}$ defined as above. That is, if we choose σ to be linear then the complicated shallow FNN $f : \mathbb{R} \to \mathbb{R}$ in (1.3) is just another linear function itself. All the weight and bias parameters used to define it were a total waste of time! Stated another way, choosing σ to be linear only allows shallow FNNs such as (1.3) to express simple linear functions.

As we shall see next, choosing σ to be something even "barely nonlinear" such as a ReLU function $\sigma(y) = \mathrm{ReLU}(y) := \max(0, y)$ already allows shallow FNNs such as (1.3) to express/represent significantly more complicated functions than simple linear ones.[3] The following Theorem is paraphrased from Foucart's fantastic book on data science [17]. Informally, it tells us that choosing σ to be a ReLU function allows shallow FNNs such as (1.3) to express any continuous piecewise linear function you like. Note that this is a dramatically larger class of functions than the simple linear ones shallow FNNs such as (1.3) can express if σ is chosen to be linear. Hence, in this case *choosing σ to be nonlinear increases expressivity.*

Theorem 1.2.9 (See Theorem 24.1 in [17])**.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU function* $\mathrm{ReLU}(x) = \max\{0, x\}$*. Then, every continuous piecewise linear function $f : \mathbb{R} \to \mathbb{R}$ as in* (1.4) *can be expressed by a shallow FNN whose single hidden layer contains $n + 2$ neurons. More specifically, let*

$$f(x) = \begin{cases} a_0 x + b_0 & x \leq \tau_1 \\ a_1 x + b_1 & \tau_1 \leq x \leq \tau_2 \\ \qquad\vdots & \\ a_{n-1} x + b_{n-1} & \tau_{n-1} \leq x \leq \tau_n \\ a_n x + b_n & \tau_n \leq x \end{cases} \tag{1.4}$$

where $\tau_1 < \tau_2 < \cdots < \tau_n$ are real numbers, and $a_0, \ldots, a_n$ and $b_0, \ldots, b_n$ are real numbers such that the function f above is continuous (i.e., $a_j \tau_{j+1} + b_j = a_{j+1} \tau_{j+1} + b_{j+1}$ for all $j \in [N]$). In other words, f is a piecewise linear function whose slope changes finitely many (specifically, n) times. Any such function can be obtained via a shallow FNN of width $n + 2$.

[3] Note that the ReLU function itself is linear everywhere except at 0. Hence, I feel it is appropriate to label it as "barely nonlinear".

Proof. We begin by noting two useful properties of the ReLU function:

$$\begin{aligned}&\mathrm{ReLU}(\gamma x) = \gamma \mathrm{ReLU}(x)\ \forall x \in \mathbb{R}, \gamma > 0, \quad \text{and}\\ &x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x)\ \forall x \in \mathbb{R}.\end{aligned}$$

Using these two properties, we can write f as the following linear combination of $n+2$ ReLU functions as follows

$$\begin{aligned}f(x) &= a_0 x + b_0 + \sum_{j=1}^{n} (a_j - a_{j-1}) \mathrm{ReLU}(x - \tau_j)\\ &= \mathrm{ReLU}(a_0 x + b_0) - \mathrm{ReLU}(-a_0 x - b_0) + \sum_{j=1}^{n} (a_j - a_{j-1}) \mathrm{ReLU}(x - \tau_j).\end{aligned}$$

□

Note that the class of piecewise linear functions is actually quite powerful approximation-theoretically since one can, e.g., approximate any continuous function $\mathbb{R} \to \mathbb{R}$ within a bounded domain arbitrarily well using increasingly fine piecewise linear approximations. Thus, the theorem above tells us that even when using the most basic tools available to us (a straightforward *nonlinear* activation function within a FNN with just a single layer) we can already approximate a very general class of functions from $\mathbb{R} \to \mathbb{R}$ as well as we want.

When we consider functions of two variables, however, things become a bit more complicated. For example, [17] also shows that the bivariate piecewise linear function $g(x_0, x_1) = \min(0, \max(x_0, x_1))$ can *not* be exactly represented by a *shallow* ReLU FNN of any width. That said, as the next theorem demonstrates, g can in fact be exactly represented by a FNN of depth $L = 2$. This simple example is meant to demonstrate the following more general principal: *Increasing the depth of a FNN increases its expressivity.* In practice the depths (and widths) of modern neural networks are very large for this reason, leading to the necessity of practitioners to deal with many very large matrices. This is just one of the many many reasons it's crucial for the modern data scientist to know the linear algebra we will review in the next chapter. Hope to see you there!

Theorem 1.2.10 (Section 24.3 in [17])**.** *Define the function* $g : \mathbb{R}^2 \to \mathbb{R}$ *by* $g(x_0, x_1) = \min\{0, \max\{x_0, x_1\}\}$*. This function* g *cannot be generated by a shallow ReLU FNN, but* g *can be obtained as a depth* $L = 2$ *ReLU FNN.*

Proof. For a proof that g cannot be generated by a shallow ReLU FNN, consult [17, Theorem 24.1]. Below we show explicitly how g can be written as a depth 2 ReLU FNN.

$$\begin{aligned}g(x_0, x_1) &= \min\{0, \max\{x_0, x_1\}\}\\ &= -\mathrm{ReLU}(-\max\{x_0, x_1\})\\ &= -\mathrm{ReLU}(-(x_0 + \mathrm{ReLU}(x_1 - x_0)))\\ &= -\mathrm{ReLU}(-\mathrm{ReLU}(x_0) + \mathrm{ReLU}(-x_0) - \mathrm{ReLU}(x_1 - x_0))\end{aligned}$$

We can also draw this neural network as in Figure 1.7, omitting arrows with weight 0. □

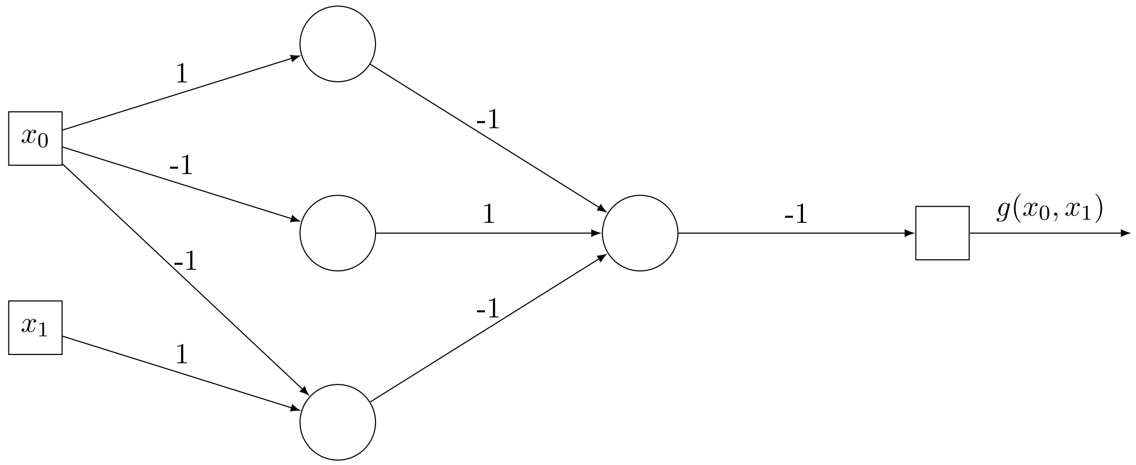

Figure 1.7: The graphical representation of the depth $L = 2$ ReLU FNN from the proof of Theorem 1.2.10 that computes $g(x_0, x_1) = \min\{0, \max\{x_0, x_1\}\}$.

Chapter 2

Linear Algebra over the Real and Complex Numbers

In this chapter we will introduce/review linear algebra over the complex numbers. We note immediately, however, that the real numbers are also complex numbers! **If the reader is intimidated by (or temporarily disinterested in) doing linear algebra over the complex numbers, they can simply skip down to Section 2.2 and replace the symbol "$\mathbb{C}$" everywhere it appears there with an "$\mathbb{R}$". Doing so will not affect the correctness of anything in this chapter, or limit your understanding in an important way until Section 3.7.** We will also continue to use the matrix notation and conventions discussed in, e.g., Section 1.2.2 going forward. All of that material (where one restricts oneself to thinking about the reals $\mathbb{R} \subset \mathbb{C}$) also remains true in this chapter. In short, if you know how linear algebra works over $\mathbb{C}$, then you can reduce to linear algebra over $\mathbb{R}$ by simply replacing "$\mathbb{C}$" everywhere it appears with an "$\mathbb{R}$". Doing linear algebra over the complex numbers instead of the reals in the first place does require a few minor adaptations, though (mainly, you need to use complex conjugation in a few crucial definitions). We will do that for you below. Before we begin, however, let's review the complex numbers.

2.1 The Complex Numbers

In this book the letter $\mathbb{i}$ will be reserved for the imaginary number $\sqrt{-1}$. That is, $\mathbb{i}^2 := -1$. The imaginary number $\mathbb{i}$ satisfies all the properties you hope it would when interacting with elements of $\mathbb{R}$ including: $0\mathbb{i} = 0$ and $1\mathbb{i} = \mathbb{i}$, as well as all the usual associative, commutative, and distributive properties (e.g., $\mathbb{i}x = x\mathbb{i}$ and $\mathbb{i} + x = x + \mathbb{i}$ $\forall x \in \mathbb{R}$). A **complex number** is an object of the form $z = x + \mathbb{i}y$ for $x, y \in \mathbb{R}$. The set of complex numbers is denoted

$$\mathbb{C} := \{x + \mathbb{i}y \mid x, y \in \mathbb{R}\}.$$

The number x in $z = x + \mathbb{i}y$ is called the real part of z, and is denoted $\text{Re}(z) \in \mathbb{R}$. Similarly, the number y is called the imaginary part of z, and is denoted $\text{Im}(z) \in \mathbb{R}$. A real number is simply a complex number with a zero imaginary part. Hence, $\mathbb{R} \subset \mathbb{C}$. There is also a common geometric interpretation of a complex number as illustrated in Figure 2.1. In fact, the existence of this picture is why $\mathbb{C}$ is sometimes also referred to as "the complex plane".

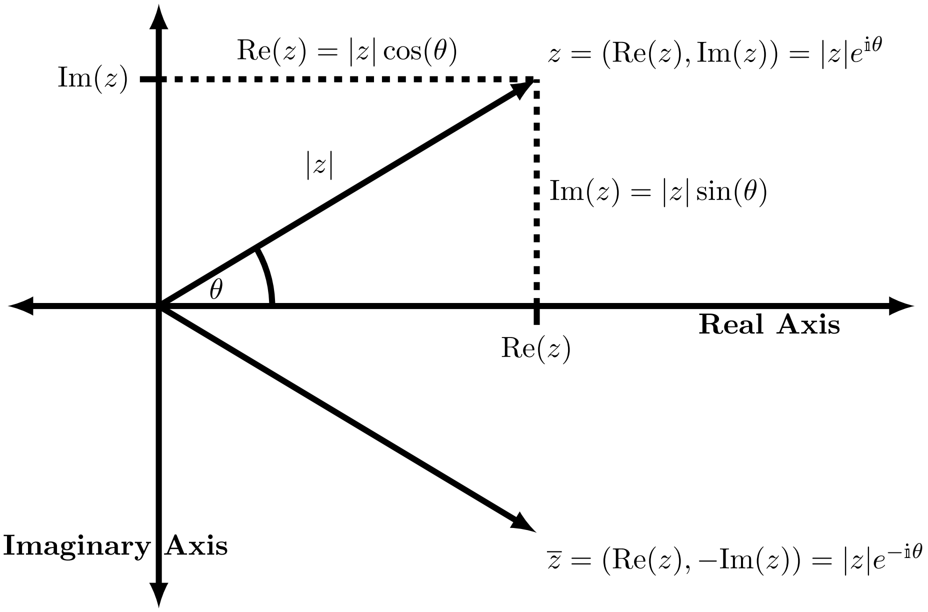
$\text{Re}(z) = |z|\cos(\theta)$
$z = (\text{Re}(z), \text{Im}(z)) = |z|e^{\mathbb{i}\theta}$
$\text{Im}(z)$
$|z|$
$\text{Im}(z) = |z|\sin(\theta)$
θ
Real Axis
$\text{Re}(z)$
Imaginary Axis
$\overline{z} = (\text{Re}(z), -\text{Im}(z)) = |z|e^{-\mathbb{i}\theta}$

Figure 2.1: The geometry of a complex number $z \in \mathbb{C}$.

Figure 2.1 represents many of most important quantities related to a complex number $z = x + \mathbb{i}y$ stemming from geometry. In particular, the **modulus**, **magnitude**, or **absolute value** of $z = x + \mathbb{i}y$ is denoted by $|z|$. It is defined to be the Euclidean distance from the origin to $(\text{Re}(z), \text{Im}(z)) = (x, y)$ in the complex plane. It is therefore also the length of the hypotenuse of a right triangle whose other two sides have lengths $|\text{Re}(z)|$ and $|\text{Im}(z)|$, and so can be computed using the Pythagorean theorem to be

$$|z| = \sqrt{(\text{Re}(z))^2 + (\text{Im}(z))^2} = \sqrt{x^2 + y^2}.$$

Note that if $z \in \mathbb{R}$ so that $z = \text{Re}(z)$ (i.e., if $y = 0$) then $|z| = |\text{Re}(z)| = |x|$. That is, this definition extends the usual definition of absolute value over the real numbers $\mathbb{R}$ to all of $\mathbb{C}$.

Exercise 2.1.1. *Let $z \in \mathbb{C}$. Prove that $|\text{Re}(z)| \leq |z|$ and $|\text{Im}(z)| \leq |z|$ always hold.*

Another fundamental geometric quantity illustrated in Figure 2.1 related to $z = x + \mathbb{i}y$ is its **phase angle** or **argument**, $\theta = \arg(z) \in [0, 2\pi)$, defined to be the angle between the real axis and the vector from the origin to $(\text{Re}(z), \text{Im}(z)) = (x, y)$ in the complex plane.

Using the geometric definitions of sin and cos involving right triangles one can immediately derive the formulas

$$x = \mathrm{Re}(z) = |z|\cos\theta \quad \text{and} \quad y = \mathrm{Im}(z) = |z|\sin\theta.$$

Similarly, one can appeal to trigonometry to see that, e.g, the phase angle θ of $z = x + \mathbb{i}y$ is

$$\theta = \arg(z) := \cos^{-1}\left(\frac{\mathrm{Re}(z)}{|z|}\right) = \cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2}}\right),$$

where one needs to remember to correct θ based on the quadrant of the complex plane z belongs to in the usual way. Note that positive real numbers (with sign $1 = \cos(0)$) always have the phase angle $\theta = 0$, and that negative real numbers (with sign $-1 = \cos(\pi)$) always have the phase angle $\theta = \pi$. Hence, phase angles effectively extend the notion of "sign" from the real numbers $\mathbb{R}$ to all of $\mathbb{C}$ in a consistent fashion.

Two complex numbers $z_1 = x_1 + \mathbb{i}y_1$ and $z_2 = x_2 + \mathbb{i}y_2$ can be added component-wise (effectively as vectors) via the definition

$$z_1+z_2 = (x_1+\mathbb{i}y_1)+(x_2+\mathbb{i}y_2) := (x_1+x_2)+\mathbb{i}(y_1+y_2) = \mathrm{Re}(z_1)+\mathrm{Re}(z_2)+\mathbb{i}(\mathrm{Im}(z_1)+\mathrm{Im}(z_2)).$$

Note again that the usual relationship between $\mathbb{R}$ and $\mathbb{C}$ holds: if $z_1, z_2 \in \mathbb{R}$ so that $\mathrm{Im}(z_1) = \mathrm{Im}(z_2) = 0$ then this definition of addition matches addition in $\mathbb{R}$. We have once again managed to extend the usual definition (of addition here) from $\mathbb{R}$ to all of $\mathbb{C}$ in a totally consistent way.

Similarly, two complex numbers $z_1, z_2 \in \mathbb{C}$ can be multiplied using the standard distributive law for the multiplication of two real numbers, but making sure to use the identity $\mathbb{i}^2 = -1$. Indeed, if $z_1 = x_1 + \mathbb{i}y_1$ and $z_2 = x_2 + \mathbb{i}y_2$, then

$$\begin{aligned} z_1z_2 = (x_1 + \mathbb{i}y_1)(x_2 + \mathbb{i}y_2) &:= x_1x_2 + \mathbb{i}x_1y_2 + \mathbb{i}y_1x_2 + \mathbb{i}^2y_1y_2 \\ &= (x_1x_2 - y_1y_2) + \mathbb{i}(x_1y_2 + y_1x_2). \end{aligned}$$

Note once again that this definition of multiplication matches multiplication over the reals whenever $z_1, z_2 \in \mathbb{R}$ so that $y_1 = y_2 = 0 = \mathrm{Im}(z_1) = \mathrm{Im}(z_2)$. This, of course, allows us to compute powers of $z \in \mathbb{C}$, z^n, for any positive integer n in a way that is again a consistent extension of how one computes powers of real numbers.

Exercise 2.1.2. *Verify the following properties of complex number addition and multiplication.*

1. **Commutativity of addition and multiplication**: $z_1+z_2 = z_2+z_1$ *and* $z_1z_2 = z_2z_1$ *for all* $z_1, z_2 \in \mathbb{C}$.

2. **Associativity of addition and multiplication**: $(z_1 + z_2) + z_3 = z_1 + (z_2 + z_3)$ *and* $(z_1z_2)z_3 = z_1(z_2z_3)$ *for all* $z_1, z_2, z_3 \in \mathbb{C}$.

3. **Distributivity***:* $z_1(z_2 + z_3) = z_1z_2 + z_1z_3$ *for all* $z_1, z_2, z_3 \in \mathbb{C}$*.*

Exercise 2.1.3. *Let* $z_1, z_2 \in \mathbb{C}$*. Show that* $|z_1z_2| = |z_1||z_2|$*.*

Importantly, we can now see that more complicated functions that can be defined on $\mathbb{R}$ in terms of series expansions (like exp, cos, sin, ...) should also believably extend in a consistent way to all of $\mathbb{C}$ since all of their basic building blocks (addition, multiplication, and integer powers) have been consistently extended from $\mathbb{R}$ to all of $\mathbb{C}$.[1] Recall the Taylor series for the exponential function centered at 0 is

$$\exp(x) = \mathbb{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

This series converges absolutely for every $x \in \mathbb{R}$. The exponential function of any complex number $z \in \mathbb{C}$ can be defined analogously as

$$\exp(z) = \mathbb{e}^z = \sum_{n=0}^{\infty} \frac{z^n}{n!}. \tag{2.1}$$

It matches the usual definition of exp on $\mathbb{R}$ (i.e., whenever $z \in \mathbb{R} \subset \mathbb{C}$) for all the reasons emphasized above. For more of this extensions interesting properties on $\mathbb{C}$ we recommend taking a look at, e.g., [10].

2.1.1 Euler's Identity

We may now derive Euler's identity for complex exponentials. Consider the purely imaginary number $z = \mathbb{i}\theta$ for some $\theta \in \mathbb{R}$. In this case, we have

$$\exp(\mathbb{i}\theta) = \mathbb{e}^{\mathbb{i}\theta} = \sum_{n=0}^{\infty} \frac{(\mathbb{i}\theta)^n}{n!}. \tag{2.2}$$

Before simplifying the above expression, we observe that since $\mathbb{i}^2 = -1$, we have

$$\mathbb{i}^{2n} = (-1)^n \quad \text{and} \quad \mathbb{i}^{2n+1} = (-1)^n \mathbb{i} \quad \text{for all } n \geq 0.$$

[1] If you want to learn about how very nicely this idea ends up working out, I strongly recommend taking a class on complex analysis!

Breaking the sum (2.2) into the parts where n is even and n is odd we get that

$$\begin{aligned}
\mathbb{e}^{\mathbb{i}\theta} &= \sum_{n \text{ even}} \frac{(\mathbb{i}\theta)^n}{n!} + \sum_{n \text{ odd}} \frac{(\mathbb{i}\theta)^n}{n!} \\
&= \sum_{n=0}^{\infty} \frac{(\mathbb{i}\theta)^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \frac{(\mathbb{i}\theta)^{2n+1}}{(2n+1)!} \\
&= \sum_{n=0}^{\infty} \frac{\mathbb{i}^{2n}\theta^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \frac{\mathbb{i}^{2n+1}\theta^{2n+1}}{(2n+1)!} \\
&= \sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \frac{(-1)^n\mathbb{i}\theta^{2n+1}}{(2n+1)!} \\
&= \left(\sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n}}{(2n)!}\right) + \mathbb{i}\left(\sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n+1}}{(2n+1)!}\right).
\end{aligned}$$

Now recall from calculus that the Taylor series for $\cos\theta$ and $\sin\theta$ about 0 are

$$\cos\theta = \sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n}}{(2n)!} \quad \text{and} \quad \sin\theta = \sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n+1}}{(2n+1)!},$$

and that the series above converge absolutely for all $\theta \in \mathbb{R}$. Consequently, we obtain **Euler's identity**, i.e., that

$$\mathbb{e}^{\mathbb{i}\theta} = \cos\theta + \mathbb{i}\sin\theta. \tag{2.3}$$

Exercise 2.1.4. *Use Euler's identity and trigonometric identities involving sine and cosine to show that* $\mathbb{e}^{\mathbb{i}\theta}\mathbb{e}^{\mathbb{i}\omega} = \mathbb{e}^{\mathbb{i}(\theta+\omega)}$ *holds for all* $\omega, \theta \in \mathbb{R}$.

Exercise 2.1.5. *Use induction in addition to the last exercise to prove that* $(\mathbb{e}^{\mathbb{i}\theta})^n = \mathbb{e}^{\mathbb{i}n\theta}$ *holds for all* $n \in \mathbb{N}$.

2.1.2 The Polar Representation of a Complex Number

As illustrated in Figure 2.1, every $z = x + \mathbb{i}y \in \mathbb{C}$ corresponds to a point $(x, y) = (\mathrm{Re}(z), \mathrm{Im}(z))$ in the complex plane. This suggests another way of representing a complex number using polar coordinates as done for $\mathbb{R}^2$. Specifically, if $x = r\cos\theta$ and $y = r\sin\theta$ for some $r \geq 0$ and $\theta \in [0, 2\pi)$, then a complex number $z = x + \mathbb{i}y$ can be represented in terms of r and θ as follows:

$$z = x + \mathbb{i}y = r\cos\theta + \mathbb{i}r\sin\theta = r\left(\cos\theta + \mathbb{i}\sin\theta\right). \tag{2.4}$$

Note that the identity $\cos^2(\theta) + \sin^2(\theta) = 1$ shows that $r = |z|$ in the polar representation above.

Using Euler's identity in (2.4) gives us the polar representation of a complex number in terms of complex exponentials:

$$z = r\mathbb{e}^{\mathbb{i}\theta}.$$

Stating the same formula another way, we have that

$$z = \mathrm{Re}(z) + \mathbb{i}\mathrm{Im}(z) = |z|\mathbb{e}^{\mathbb{i}\arg(z)}.$$

Exercise 2.1.6. *Prove the following useful identities involving the polar representation of complex numbers.*

1. *Show that if $z = r\mathbb{e}^{\mathbb{i}\theta}$, then for any $n \in \mathbb{N}$ we have $z^n = r^n\left(\cos(n\theta) + \mathbb{i}\sin(n\theta)\right).$*

2. *Every complex number of unit modulus can be written as $\mathbb{e}^{\mathbb{i}\theta}$ for some $\theta \in [0, 2\pi)$.*

3. *If $z = r\mathbb{e}^{\mathbb{i}\theta}$ and $w = s\mathbb{e}^{\mathbb{i}\varphi}$, then $zw = rs\mathbb{e}^{\mathbb{i}(\theta+\varphi)}$.*

2.1.3 Complex Conjugation

The **complex conjugate** of a complex number $z = x + \mathbb{i}y$ is the complex number $\bar{z} = x - \mathbb{i}y$. Geometrically, as illustrated in Figure 2.1, $\bar{z}$ is the reflection of z across the real axis. One can verify that

$$\mathrm{Re}(z) = \frac{z + \bar{z}}{2}, \quad \mathrm{Im}(z) = \frac{z - \bar{z}}{2\mathbb{i}}.$$

Consequently, a complex number z is a real number if and only if $z = \bar{z}$.

Exercise 2.1.7. *Prove the following useful identities involving complex conjugation. Let $z_1, z_2 \in \mathbb{C}$.*

1. *Show that $\overline{z_1 + z_2} = \overline{z_1} + \overline{z_2}$.*

2. *Show that $\overline{z_1 z_2} = \overline{z_1}\ \overline{z_2}$.*

3. *Show that $|z_1|^2 = z_1\overline{z_1}$.*

4. *Show that $|z_1 + z_2|^2 = |z_1|^2 + |z_2|^2 + 2\mathrm{Re}(z_1\overline{z_2})$.*

Exercise 2.1.8. *Let $z \in \mathbb{C}$ have the polar representation $z = r\mathbb{e}^{\mathbb{i}\theta}$. Show that $\bar{z} = r\mathbb{e}^{-\mathbb{i}\theta}$.*

Exercise 2.1.9. *Let $z \in \mathbb{C}$ be nonzero. Show that*

$$z^{-1} := \frac{1}{z} = \frac{\bar{z}}{|z|^2}.$$

2.1.4 The Roots of Unity

Fix $n \in \mathbb{N}$ and consider the equation

$$z^n = 1, \quad z \in \mathbb{C}.$$

We wish to find all solutions $z \in \mathbb{C}$ of this equation. First, notice that necessarily $|z| = 1$. Hence, $z = \mathbb{e}^{\mathbb{i}\theta}$ for $\theta \in [0, 2\pi)$. By Euler's formula, this means that

$$1 = \cos(n\theta) + \mathbb{i}\sin(n\theta),$$

which implies $\cos(n\theta) = 1$ and $\sin(n\theta) = 0$. This can only happen if $n\theta = 2\pi k$ for some $k \in \mathbb{Z}$. Thus, $\theta = \frac{2\pi k}{n}$ must hold. Note that we have n distinct values of $\theta \in [0, 2\pi)$ satisfying this formula, one for each $k \in [n]$. The set of these solutions is therefore $\{\mathbb{e}^{\frac{2\pi k \mathbb{i}}{n}} \mid k \in [n]\}$ are called the n^{th} **roots of unity**. Notice that they all satisfy $z^n = 1$ by design, and are placed in an equidistant fashion around the unit circle $|z| = 1$ in the complex plane. These values will be of special significance later in Section 3.7.

2.1.5 The Triangle Inequality for Complex Numbers

We will now prove the triangle inequality for complex numbers.

Lemma 2.1.1 (The Triangle Inequality for $\mathbb{C}$)**.**

$$|z_1 + z_2| \leq |z_1| + |z_2| \qquad \textit{for all } z_1, z_2 \in \mathbb{C}. \tag{2.5}$$

Furthermore, equality holds in (2.5) *if and only if* $z_1 = cz_2$ *for some real number* $c \geq 0$*.*

Proof. Using the results of Exercises 2.1.7, 2.1.1, and 2.1.3 we can see that

$$\begin{aligned}
|z_1 + z_2|^2 &= |z_1|^2 + |z_2|^2 + 2\mathrm{Re}(z_1\overline{z_2}) \\
&\leq |z_1|^2 + |z_2|^2 + 2|z_1\overline{z_2}| \\
&= |z_1|^2 + |z_2|^2 + 2|z_1||\overline{z_2}| \\
&= |z_1|^2 + |z_2|^2 + 2|z_1||z_2| \\
&= (|z_1| + |z_2|)^2 .
\end{aligned}$$

Taking square roots now gives us the desired inequality.

Now suppose that we have equality in (2.5). If either z_1 or z_2 is 0 we are finished. Thus, suppose that $z_1 = r_1\mathbb{e}^{\mathbb{i}\theta_1}$ and $z_2 = r_2\mathbb{e}^{\mathbb{i}\theta_2}$ with $r_1, r_2 > 0$. Notice that the only place we have an inequality in the argument above is in the estimate $\mathrm{Re}(z_1\overline{z_2}) \leq |z_1\overline{z_2}|$. If $|z_1 + z_2| = |z_1| + |z_2|$, then we must, in fact, have $\mathrm{Re}(z_1\overline{z_2}) = |z_1\overline{z_2}|$. That means $\mathrm{Im}(z_1\overline{z_2}) = r_1 r_2 \sin(\theta_1 - \theta_2) = 0$. Given that $r_1, r_2 > 0$ we must therefore have $\sin(\theta_1 - \theta_2) = 0$, implying that $\theta_1 - \theta_2 = m\pi$ for some integer m.

If m were odd it would imply that

$$\mathrm{Re}(z_1\overline{z_2}) = r_1r_2\cos(\theta_1-\theta_2) = r_1r_2\cos(m\pi) = -r_1r_2 < 0.$$

This is impossible here since we have $\mathrm{Re}(z_1\overline{z_2}) = |z_1z_2| = r_1r_2 > 0$. Therefore, m must be even, and so $\theta_1-\theta_2 = 2\pi n$ for some integer n. This implies that $\arg(z_1) = \arg(z_2)$, and so $z_1 = cz_2$ for some positive real number c. □

We now have all the prerequisites we need to begin discussing linear algebra over $\mathbb{C}$.

2.2 Basic Linear Algebra over $\mathbb{C}$ and $\mathbb{R}$

A complex valued matrix $A \in \mathbb{C}^{m\times n}$ is a matrix of complex values with m rows and n columns whose entries are denoted by $A_{j,k} \in \mathbb{C}$ for all $j \in [m]$ and $k \in [n]$. A complex valued vector $\mathbf{x} \in \mathbb{C}^n$ of length n is also considered to be an $n \times 1$ matrix (i.e, vectors are "column vectors" by default). It's entries are denoted by $x_j \in \mathbb{C}$ for all $j \in [n]$, and can themselves be safely considered to be scalars, length 1 vectors, and 1×1 matrices as convenient. Matrices (and vectors) are always added entrywise, and scalar-vector/scalar-matrix multiplication is also always performed entrywise, as in your first linear algebra course.

Given a matrix $A \in \mathbb{C}^{m\times n}$ and a vector $\mathbf{x} \in \mathbb{C}^n$, their **matrix-vector product**, $A\mathbf{x} \in \mathbb{C}^m$, is a vector which can be defined in two equivalent ways. First, it can be defined entrywise via

$$(A\mathbf{x})_j := \sum_{k\in[n]} A_{j,k}x_k \in \mathbb{C} \quad \forall j \in [m]. \tag{2.6}$$

Alternatively, it can defined as a weighted sum of the columns of A via the formula

$$A\mathbf{x} = \sum_{k\in[n]} x_kA_{:,k} \in \mathbb{C}^m. \tag{2.7}$$

Both equations are true and will be used often below.

Exercise 2.2.1. *Let $j \in [n]$. Show that the function $(\cdot)_j : \mathbb{C}^n \to \mathbb{C}$ that maps a vector to it's j^{th} entry is a linear function (i.e., argue that $(\alpha\mathbf{x}+\beta\mathbf{y})_j = \alpha(\mathbf{x})_j + \beta(\mathbf{y})_j$ holds for all $\alpha,\beta \in \mathbb{C}$ and $\mathbf{x},\mathbf{y} \in \mathbb{C}^n$).*

Exercise 2.2.2. *Show that* (2.6) *holds if and only if* (2.7) *holds.*

We can use the matrix-vector product notation to describe the **product of two matrices**. For any natural numbers m, n, p, and two matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times p}$, we can define their product columnwise by $(AB)_{:,k} := A(B_{:,k}) = A\mathbf{b}_k \; \forall k \in [p]$, where

$\mathbf{b}_k = B_{:,k}$ denotes the k^{th} column of B, and $A\mathbf{b}_k$ is a matrix-vector product as per (2.6). Equivalently we may write

$$AB = A \begin{pmatrix} | & & | \\ \mathbf{b}_0 & \cdots & \mathbf{b}_{p-1} \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ A\mathbf{b}_0 & \cdots & A\mathbf{b}_{p-1} \\ | & & | \end{pmatrix}. \tag{2.8}$$

Similarly, we may also define matrix-matrix multiplication entrywise by

$$(AB)_{j,k} \ := \ \sum_{l=0}^{n-1} A_{j,l} B_{l,k}. \tag{2.9}$$

Note further that since we always consider a vector $\mathbf{v} \in \mathbb{C}^n$ to be an $n \times 1$ matrix, we should check that the resulting matrix-matrix product $A\mathbf{v}$ agrees with the matrix-vector product definition of $A\mathbf{v}$ above. It does – check!

Exercise 2.2.3. *Show that* (2.8) *holds if and only if* (2.9) *holds by first verifying that* $(A(B_{:,k}))_j \ = \ \sum_{l=0}^{n-1} A_{j,l} B_{l,k}$.

Matrix-vector multiplication further allows us to view matrices as functions. Given $A \in \mathbb{C}^{m \times n}$, A acts on $\mathbb{C}^n$ by vector multiplication, and can therefore be viewed as a map $A : \mathbb{C}^n \to \mathbb{C}^m$ defined by $A(\mathbf{x}) = A\mathbf{x}$ for all $\mathbf{x} \in \mathbb{C}^n$. One can confirm that A is then a linear function, and that the range of A (as a function) is the column space of A (as a matrix). That is,

$$\begin{aligned} \text{Range}(A) :=& \ \{A\mathbf{x} \mid \mathbf{x} \in \mathbb{C}^n\} \subset \mathbb{C}^m \\ =& \ \left\{ \sum_{j \in [n]} \alpha_j A_{:,j} \mid \alpha_j \in \mathbb{C} \ \forall j \in [n] \right\} \ =: \ \text{span}\left(\{A_{:,j} \mid j \in [n]\}\right) \\ =:& \ \mathcal{C}(A) \ = \ \text{Column Space of } A. \end{aligned}$$

Let $A \in \mathbb{C}^{m \times n}$ be a matrix. The **adjoint** of A, denoted $A^* \in \mathbb{C}^{n \times m}$, is the conjugate transpose of A, i.e., the matrix produced by transposing A and taking the complex conjugate of each entry. It is defined entrywise by $(A^*)_{j,k} = \overline{A_{k,j}}$. Note that if $A \in \mathbb{R}^{m \times n}$ then $A^* = A^T$. We also note that $A = (A^*)^*$ always holds (check this!).

Exercise 2.2.4. *Let* $A, B \in \mathbb{C}^{m \times n}$. *Show that* $(A + B)^* = A^* + B^*$.

Exercise 2.2.5. *Let* $A \in \mathbb{C}^{m \times n}$ *and* $B \in \mathbb{C}^{n \times p}$. *Show that* $(AB)^* = B^* A^*$.

Given two vectors of the same length, $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, we can define their Euclidean **inner product** to be

$$\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{j=0}^{n-1} \overline{x_j} y_j = \mathbf{x}^* \mathbf{y} \in \mathbb{C}.$$

Also note that when two vectors $\mathbf{x}$ and $\mathbf{y}$ are real-valued, the complex inner product of $\mathbf{x}$ and $\mathbf{y}$ equals the real inner product of $\mathbf{x}$ and $\mathbf{y}$. Thus, we can view linear algebra over the complex numbers as a natural extension of linear algebra over the reals, where any statement about complex linear algebra still holds true when we restrict ourselves to the real numbers. This again supports my prior claim that you can simply "replace $\mathbb{C}$ everywhere in this section with $\mathbb{R}$" and have a chapter on linear algebra over $\mathbb{R}$ as a result, should you desire to do so.

These next four exercises are highly recommended. As always, using the result of prior exercises to will help you complete subsequent ones more quickly is also always highly recommended.

Exercise 2.2.6. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$. Show that $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$.*

Exercise 2.2.7. *Show that the inner product is conjugate-linear in the first argument and linear in the second argument. That is, for $\alpha, \beta \in C$ and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{C}^n$ show that*

1. $\langle \alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z} \rangle = \overline{\alpha}\langle \mathbf{x}, \mathbf{z} \rangle + \overline{\beta}\langle \mathbf{y}, \mathbf{z} \rangle$*, and that*

2. $\langle \mathbf{x}, \alpha\mathbf{y} + \beta\mathbf{z} \rangle = \alpha\langle \mathbf{x}, \mathbf{y} \rangle + \beta\langle \mathbf{x}, \mathbf{z} \rangle$.

Exercise 2.2.8. *Let $A \in \mathbb{C}^{m\times n}$ and $\mathbf{x} \in \mathbb{C}^n$. Show that $(A\mathbf{x})_j = \langle (A^*)_{:,j}, \mathbf{x} \rangle = \langle \overline{A_{j,:}}, \mathbf{x} \rangle$ for all $j \in [m]$.*

Exercise 2.2.9. *Let $A \in \mathbb{C}^{m\times n}$, $\mathbf{x} \in \mathbb{C}^n$, and $\mathbf{y} \in \mathbb{C}^m$. Show that both $\langle A\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, A^*\mathbf{y} \rangle$ and $\langle A^*\mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{y}, A\mathbf{x} \rangle$ hold.*

Let's now briefly review a geometric concept related to inner products that's reserved for real-valued vectors.

2.2.1 Some Inner Product Geometry for Real-valued Vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

The inner product can be used to express the angle between two real vectors. Given two non-zero vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the **angle** $\theta \in [0, \pi]$ **between** $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is

$$\theta = \cos^{-1}\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\sqrt{\langle \mathbf{x}, \mathbf{x} \rangle\langle \mathbf{y}, \mathbf{y} \rangle}}\right). \tag{2.10}$$

Note that $\theta = \pi/2$ (or 90 degrees) whenever $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, indicating that the two vectors are perpendicular, or orthogonal, to one another. Further note that the angle between $\mathbf{x}$ and $\mathbf{y}$ can always be reasoned about with regular two-dimensional plane geometry no matter how large n is here since $\mathbf{x}$ and $\mathbf{y}$ will always belong to the (at most) two-dimensional subspace $\mathrm{span}\{\mathbf{x}, \mathbf{y}\} \subset \mathbb{R}^n$. Hence, all the pictures of right triangles you are tempted to draw on a piece of paper to better understand θ are 100% justified.[2]

Back to $\mathbb{C}^n$: Using the inner product geometry for real-valued vectors as motivation, we will also say that **two complex-valued vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ are orthogonal** if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. We will now recall an important inequality for inner products.

2.2.2 The Cauchy–Schwarz Inequality

Note that for any vector $\mathbf{x} \in \mathbb{C}^n$, $\langle \mathbf{x}, \mathbf{x} \rangle = \sum_{j \in [n]} x_j \overline{x_j} = \sum_{j \in [n]} |x_j|^2 \geq 0$ (this fact will become important later). Now let $t \in \mathbb{R}$, and $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, and set $\alpha := \frac{|\langle \mathbf{y}, \mathbf{x} \rangle|}{\langle \mathbf{y}, \mathbf{x} \rangle}$. One can see that α is a complex number with magnitude 1. Finally, define the function $f : \mathbb{R} \to \mathbb{R}$ by

$$f(t) := \langle t\alpha\mathbf{x} + \mathbf{y}, t\alpha\mathbf{x} + \mathbf{y} \rangle$$

Recall that $f(t) \geq 0$ for all $t \in \mathbb{R}$ by the fact above.

Continuing, the following sequence of inequalities can be seen to hold using properties of the inner product together with the definition of α (check each step!). We have that

$$\begin{aligned} 0 \leq f(t) &= t\overline{\alpha}\langle \mathbf{x}, t\alpha\mathbf{x} + \mathbf{y} \rangle + \langle \mathbf{y}, t\alpha\mathbf{x} + \mathbf{y} \rangle \\ &= t^2 \overline{\alpha}\alpha \langle \mathbf{x}, \mathbf{x} \rangle + t\overline{\alpha}\langle \mathbf{x}, \mathbf{y} \rangle + t\alpha \langle \mathbf{y}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle \\ &= t^2 \langle \mathbf{x}, \mathbf{x} \rangle + 2\mathrm{Re}(t\alpha\langle \mathbf{y}, \mathbf{x} \rangle) + \langle \mathbf{y}, \mathbf{y} \rangle \\ &= \langle \mathbf{x}, \mathbf{x} \rangle t^2 + 2|\langle \mathbf{x}, \mathbf{y} \rangle| t + \langle \mathbf{y}, \mathbf{y} \rangle, \end{aligned}$$

which is a quadratic polynomial in t with real coefficients. Since the polynomial f above is ≥ 0 for all t, it must have at most one real root.

Recalling the quadratic equation for a generic polynomial $p(t) = at^2 + bt + c$, we note that its discriminant $b^2 - 4ac$ must be non-positive (i.e., ≤ 0) in order for the polynomial to have at most one real root. Applying this to our f above we learn that

$$\begin{aligned} (2|\langle \mathbf{x}, \mathbf{y} \rangle|)^2 &\leq 4\langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle \\ |\langle \mathbf{x}, \mathbf{y} \rangle| &\leq \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle}. \end{aligned}$$

This inequality holds for all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ since we chose them arbitrarily. It is known as the Cauchy-Schwarz Inequality (i.e., it has a name!) due to its importance.

Lemma 2.2.1 (The Cauchy-Schwarz Inequality). *For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$,*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle}.$$

It is expressed here slightly differently than usual in Lemma 2.2.1, however. Usually it is stated like "$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$" where $\| \cdot \|_2$ denotes the ℓ^2-vector norm, which we will recall next.

[2] Simply rewrite $\mathbf{x}$ and $\mathbf{y}$ in terms of an orthonormal basis of the span$\{\mathbf{x}, \mathbf{y}\}$, and then draw your pictures with axes in the directions of these orthonormal basis vectors. If this footnote is confusing I recommend you continue on, review orthogonality and orthogonal projections, and then come back here again for a rematch.

2.2.3 General Norms on $\mathbb{C}^{m\times n}$, and the Euclidean Vector Norm

A **matrix norm on** $\mathbb{C}^{m\times n}$ is a function $f : \mathbb{C}^{m\times n} \to \mathbb{R}^+ := [0,\infty)$ satisfying all of the following properties:

1. (The triangle inequality): $f(A+B) \leq f(A) + f(B)$ for all $A, B \in \mathbb{C}^{m\times n}$,
2. $f(\alpha A) = |\alpha| f(A)$ for all $\alpha \in \mathbb{C}$ and $A \in \mathbb{C}^{m\times n}$, and
3. $f(A) = 0 \iff A = 0_{m\times n}$, where $0_{m\times n}$ denotes the $m \times n$ matrix of all zeros (i.e., the zero matrix).

Recall that we also view vectors in $\mathbb{C}^m$ as $m \times 1$ matrices. Thus, a norm on $m \times 1$ matrices (i.e., on vectors in $\mathbb{C}^m$) will also be called a **vector norm** for this reason.

We can now see that the **Euclidean**, or **ℓ^2-norm**, of a vector $\mathbf{x} \in \mathbb{C}^n$ defined by $\|\mathbf{x}\|_2 := \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$ is indeed a vector norm.

Lemma 2.2.2. *Let $f : \mathbb{C}^n \to \mathbb{R}^+$ be the ℓ^2-norm so that $f(x) = \|\mathbf{x}\|_2 := \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$. We claim that $f(x) = \|\mathbf{x}\|_2$ is a vector norm on $\mathbb{C}^n$.*

Proof. We will verify that each condition of a norm is satisfied. First, we will check that the triangle inequality holds. Note that the last inequality just below depends on the Cauchy-Schwarz inequality. Let $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$. Then

$$\begin{aligned}\|\mathbf{x}+\mathbf{y}\|_2 &= \sqrt{\langle \mathbf{x}+\mathbf{y}, \mathbf{x}+\mathbf{y}, \rangle}\\ &= \sqrt{\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 + 2\mathrm{Re}(\langle \mathbf{x}, \mathbf{y}\rangle)}\\ &\leq \sqrt{\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 + 2|\langle \mathbf{x}, \mathbf{y}\rangle|}\\ &\leq \sqrt{\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 + 2\|\mathbf{x}\|_2^2\|\mathbf{y}\|_2^2}\\ &= \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2.\end{aligned}$$

Next we verify that the norm scales correctly. Let $\alpha \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^n$. We have that

$$\begin{aligned}\|\alpha\mathbf{x}\|_2 &= \sqrt{\langle \alpha\mathbf{x}, \alpha\mathbf{x}\rangle} = \sqrt{\alpha\overline{\alpha}\langle \mathbf{x}, \mathbf{x}\rangle} = \sqrt{|\alpha|^2\langle \mathbf{x}, \mathbf{x}\rangle}\\ &= |\alpha|\|\mathbf{x}\|_2.\end{aligned}$$

Finally, we verify that the ℓ^2-norm of a vector $\mathbf{x} \in \mathbb{C}^n$ can only be 0 if $\mathbf{x}$ is the vector of all zeros, $\mathbf{0}$. We have that

$$\|\mathbf{x}\|_2 = 0 \iff \|\mathbf{x}\|_2^2 = 0 \iff \sum_{j\in[n]} |x_j|^2 = 0 \iff |x_j| = 0 \;\forall j \in [n].$$

Having now shown that the ℓ^2-norm satisfies all the properties of a norm, we may conclude that it indeed is one. □

The following exercise demonstrates a useful property of the inner product which is perhaps most easily seen by using the properties of the ℓ^2-norm.

Exercise 2.2.10. *Let $\mathbf{x} \in \mathbb{C}^n$. Show that if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ for all $\mathbf{y} \in \mathbb{C}^n$, then $\mathbf{x} = \mathbf{0}$ must hold.*

Though the ℓ^2-norm is by far the most often used norm, all of the other norms in the following exercises are also commonly used. Even if you don't do each exercise (you should of course!), you should look at them for the norm definitions.

Exercise 2.2.11. *Show that the* ℓ^1**-norm** *defined by*

$$\|A\|_1 := \sum_{j \in [m], k \in [n]} |A_{j,k}|$$

is indeed a norm on $\mathbb{C}^{m \times n}$.

Exercise 2.2.12. *Show that the* **Frobenius matrix norm** *defined by*

$$\|A\|_{\mathrm{F}} := \sqrt{\sum_{j \in [m], k \in [n]} |A_{j,k}|^2}$$

is indeed a norm on $\mathbb{C}^{m \times n}$. HINT: Suppose you vectorize A. What does the Frobenius norm look like then?

Exercise 2.2.13. *Show that the* ℓ^∞**-norm** *defined by*

$$\|A\|_\infty := \max_{j \in [m], k \in [n]} |A_{j,k}|$$

is indeed a norm on $\mathbb{C}^{m \times n}$.

Exercise 2.2.14. *Show that the* (ℓ^2, ℓ^2)**-operator norm** *defined by*

$$\|A\|_{2 \to 2} := \max_{\mathbf{x} \in \mathbb{C}^n \text{ s.t. } \|\mathbf{x}\|_2 = 1} \|A\mathbf{x}\|_2$$

is indeed a norm on $\mathbb{C}^{m \times n}$.

Exercise 2.2.15. *Suppose that $f : \mathbb{C}^{m \times n} \to \mathbb{R}^+$ and $g : \mathbb{C}^{m \times n} \to \mathbb{R}^+$ are both norms on $\mathbb{C}^{m \times n}$. Let $\alpha, \beta \in \mathbb{R}^+ \setminus \{0\}$. Show that $h = \alpha f + \beta g$ will also be a norm on $\mathbb{C}^{m \times n}$.*

With the aim in mind of recalling what the "rank" of a matrix really means, let's now briefly review linear independence and subspace basis properties.

2.3 Subspaces, Span, and Linear Independence

Let $S = \{\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}\} \subset \mathbb{C}^n$ be a finite and nonempty set of vectors in $\mathbb{C}^n$. The **span** of S, denoted $\operatorname{span}(S)$, is the set

$$\operatorname{span}(S) := \left\{ \sum_{j \in [m]} \alpha_j \mathbf{v}_j \mid \alpha_0, \ldots, \alpha_{m-1} \in \mathbb{C} \right\} \subset \mathbb{C}^n.$$

If $S \subset \mathbb{C}^n$ is infinite, we instead define the span of S to be the set

$$\operatorname{span}(S) := \bigcup_{A \subset S, A \text{ finite}} \operatorname{span}(A) \subset \mathbb{C}^n.$$

Note that $S \subset \operatorname{span}(S)$ always holds for any $S \subset \mathbb{C}^n$ since $\mathbf{x} \in S$ implies that $1 \cdot \mathbf{x} \in \operatorname{span}(S)$. Furthermore, note that $\mathbf{0}$ is in the span of every nonempty set S since $\mathbf{0} = 0 \cdot \mathbf{x}$ for any $\mathbf{x} \in S$.

Exercise 2.3.1. *Verify that if $A \subset S$, then $\operatorname{span}(A) \subset \operatorname{span}(S)$.*

Exercise 2.3.2. *Let $S, T \subset \mathbb{C}^n$. Verify that $\operatorname{span}(T \cap S) \subset \operatorname{span}(T) \cap \operatorname{span}(S)$.*

A subset $\mathscr{L} \subset \mathbb{C}^n$ is called a **linear subspace of** $\mathbb{C}^n$ if $\operatorname{span}(\mathscr{L}) = \mathscr{L}$. That is, subspaces are sets that are closed under taking spans. Note that the so-called **trivial subspace** $\{\mathbf{0}\} \subset \mathbb{C}^n$ is always a subspace since $\operatorname{span}(\{\mathbf{0}\}) = \{\mathbf{0}\}$. Similarly, $\mathbb{C}^n$ is a linear subspace because both of the following hold: (i) $\mathbb{C}^n \subset \operatorname{span}(\mathbb{C}^n)$ (since $S \subset \operatorname{span}(S)$ for any $S \subset \mathbb{C}^n$), and (ii) $\operatorname{span}(\mathbb{C}^n) \subset \mathbb{C}^n$ (trivially by definition).

Lemma 2.3.1. *The span of every nonempty subset $S \subset \mathbb{C}^n$ is a linear subspace of $\mathbb{C}^n$.*

Proof. We need to show that

$$\operatorname{span}\left(\operatorname{span}(S)\right) = \operatorname{span}(S).$$

As usual with set equalities of this type we will proceed by showing that both (i) $\operatorname{span}(S) \subset \operatorname{span}\left(\operatorname{span}(S)\right)$, and (ii) $\operatorname{span}\left(\operatorname{span}(S)\right) \subset \operatorname{span}(S)$, hold. In fact (i) follows from the fact above that $S \subset \operatorname{span}(S)$ holds for any $S \subset \mathbb{C}^n$. Hence, we only really need to verify (ii).

To verify that $\operatorname{span}\left(\operatorname{span}(S)\right) \subset \operatorname{span}(S)$, let $\mathbf{y} \in \operatorname{span}\left(\operatorname{span}(S)\right)$. By the definition of span, $\mathbf{y}$ must be the linear combination of a finite number $p \in \mathbb{N}$ of elements of $\operatorname{span}(S)$. Hence, $\mathbf{y}$ will have the form

$$\mathbf{y} = \sum_{j=0}^{p-1} \beta_j \left(\sum_{k=0}^{q_j - 1} \alpha_{j,k} \mathbf{x}_{j,k} \right) = \sum_{j=0}^{p-1} \sum_{k=0}^{q_j - 1} \beta_j \alpha_{j,k} \mathbf{x}_{j,k},$$

where $\beta_j \in \mathbb{C}$ and $q_j \in \mathbb{N}$ for all $j \in [p]$, and where $\mathbf{x}_{j,k} \in S$ and $\alpha_{j,k} \in \mathbb{C}$ for all $k \in [q_j]$ for each $j \in [p]$. Thus, we can see that $\mathbf{y} \in \operatorname{span}(S)$ too since it will be a linear combination of a finite number, $\min\left(\sum_{j \in [p]} q_j, |S|\right) \in \mathbb{N}$, of elements of S. □

Given a matrix $A \in \mathbb{C}^{m\times n}$, recall that the **column space of** A, denoted by $\mathcal{C}(A)$, is defined to be $\mathcal{C}(A) := \operatorname{span}(\{A_{:,j} \mid j \in [n]\}) \subset \mathbb{C}^m$. One important consequence of Lemma 2.3.1 is that the column space of every matrix $A \in \mathbb{C}^{m\times n}$ is a linear subspace of $\mathbb{C}^m$.

Exercise 2.3.3. *Let $\mathscr{L}, \mathscr{K} \subset \mathbb{C}^n$ be two linear subspaces of $\mathbb{C}^n$. Show that $\mathscr{L} \cap \mathscr{K}$ is also a linear subspace of $\mathbb{C}^n$.*

A finite set of vectors $\{\mathbf{v}_0, \dots, \mathbf{v}_{m-1}\} \subset \mathbb{C}^n$ is called **linearly independent** if $\sum_{j\in[m]} \alpha_j \mathbf{v}_j = \mathbf{0}$ if and only if $\alpha_j = 0$ for all j. In other words, no nontrivial (i.e., all zero) linear combination of the vectors can equal the zero vector. If a set of vectors is not linearly independent, we call it **linearly dependent**.

Exercise 2.3.4. *Show that any set of vectors in $\mathbb{C}^n$ containing the zero vector is linearly dependent.*

Exercise 2.3.5. *Let $\mathbf{x} \in \mathbb{C}^n$ and suppose that $B \subset \mathbb{C}^n$ is a linearly independent set. Prove that if $\mathbf{x} \notin \operatorname{span}(B)$, then $B' = B \cup \{\mathbf{x}\}$ is a new linearly independent set with $|B| + 1$ elements.*

Definition 2.3.2 (The Standard Basis Vectors of $\mathbb{C}^n$)**.** *The* **standard basis vectors of** $\mathbb{C}^n$ *are the n vectors $\{\mathbf{e}_j\}_{j\in[n]} := \{\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{n-1}, \} \subset \mathbb{C}^n$ whose entries are given by*

$$(\mathbf{e}_j)_k = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

for all $k \in [n]$.

Example 2.3.3 (The Standard Basis Vectors are Linearly Independent)**.** *The standard basis vectors $\{\mathbf{e}_j\}_{j\in[n]} \subset \mathbb{C}^n$ are linearly independent because for any $\alpha_0, \alpha_1, \dots, \alpha_{n-1} \in \mathbb{C}$ we can see that*

$$\mathbf{0} = \sum_{j\in[n]} \alpha_j \mathbf{e}_j = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} \iff \alpha_j = 0 \ \forall j \in [n].$$

Having just defined a set of vectors called the “standard basis”, it behooves us to briefly recall what a “basis” actually is. We do so next.

2.3.1 Bases, Orthonormal Bases, Dimension, and Rank

The following lemma ultimately guarantees that the notions of “dimension” and “rank” are well defined. Since these notions are inextricably linked to the notion of a “basis”, we will prepare the ground for them here.

Lemma 2.3.4 (The Exchange Lemma)**.** *Let $B_1, B_2 \subset \mathbb{C}^n$ be finite. Furthermore, suppose that B_2 is linearly independent, and that $\mathscr{L} :=$ span$(B_2) \subset$ span(B_1). Then $|B_2| \leq |B_1|$.*

Proof. Suppose, towards a contradiction, that $|B_1| < |B_2|$. Let $B_1 = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{s-1}\} \subset \mathbb{C}^n$, and $B_2 = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{s+m-1}\} \subset \mathbb{C}^n$, where $m > 0$. Recall that the $\mathbf{y}_j$ vectors are linearly independent by assumption. Furthermore, we have the assumed inclusion $\mathscr{L} = \text{span}(B_2) \subset \text{span}\left(\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{s-1}\}\right)$.

Because $\mathbf{y}_0 \in \text{span}(B_1)$, there exist $\alpha_0, \dots, \alpha_{s-1} \in \mathbb{C}$ such that

$$\mathbf{y}_0 = \sum_{j \in [s]} \alpha_j \mathbf{x}_j.$$

Furthermore, because the $\mathbf{y}_j$ vectors are linearly independent, we recall that $\mathbf{y}_0$ can't be the zero vector. Hence, at least one of the α_j's must be nonzero. Without loss of generality (w.l.g.), we may assume that $\alpha_0 \neq 0$. Thus, we can write $\mathbf{x}_0$ in terms of $\mathbf{y}_0$ and the other $\mathbf{x}_j$'s to see that

$$\mathbf{x}_0 = \frac{1}{\alpha_0} \left(\mathbf{y}_0 - \sum_{j=1}^{s-1} \alpha_j \mathbf{x}_j \right).$$

Hence, $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{s-1}\}\right)$ also holds. Note that we have effectively exchanged $\mathbf{x}_0$ for $\mathbf{y}_0$ in our initially assumed inclusion.

Now, we repeat this process to exchange $\mathbf{x}_1$ for $\mathbf{y}_1$ in the last inclusion just above: Since $\mathbf{y}_1 \in \mathscr{L}$, $\mathbf{y}_1 \in \text{span}(\{\mathbf{y}_0, \mathbf{x}_1, \dots, \mathbf{x}_{s-1}\})$. Thus, there exists $\beta_0 \in \mathbb{C}$ and $\gamma_1, \dots, \gamma_{s-1} \in \mathbb{C}$ such that

$$\mathbf{y}_1 = \beta_0 \mathbf{y}_0 + \sum_{j=1}^{s-1} \gamma_j \mathbf{x}_j.$$

Note that at least one $\gamma_j \in \mathbb{C}$ above must be nonzero (otherwise, we'd have $\mathbf{y}_1 = \beta_0 \mathbf{y}_0$, violating the assumed linear independence of the $\mathbf{y}_j$'s). Without loss of generality, we may assume that $\gamma_1 \neq 0$. Thus, we can write

$$\mathbf{x}_1 = \frac{1}{\gamma_1} \left(\mathbf{y}_1 - \beta_0 \mathbf{y}_0 - \sum_{j=2}^{s-1} \gamma_j \mathbf{x}_j \right).$$

As a result we have successfully exchanged $\mathbf{x}_1$ for $\mathbf{y}_1$ in our prior inclusion to see that $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{y}_1, \mathbf{x}_2, \dots, \mathbf{x}_{s-1}\}\right)$ also holds.

Repeating this process $s - 2$ more times we find that $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{s-1}\}\right)$ must hold. This generates a contradiction, however, because it implies that $\mathbf{y}_s \in B_2$ can be written as a linear combination of $\mathbf{y}_0, \dots, \mathbf{y}_{s-1}$, contradicting the fact that $\mathbf{y}_j$'s are linearly independent. Therefore, $|B_1| < |B_2|$ can't hold. □

The following corollary of Lemma 2.3.4 guarantees that any two linearly independent sets that generate the same subspace have to have the same cardinality.

Corollary 2.3.5. *Let $B_1, B_2 \subset \mathbb{C}^n$ be finite sets that are both linearly independent. Furthermore, suppose that span$(B_1) =$ span(B_2). Then, $|B_1| = |B_2|$.*

Exercise 2.3.6. *Prove Corollary 2.3.5.*

We are now able to give a well defined definition of the dimension of a linear subspace. Let $\mathscr{L}$ be a linear subspace of $\mathbb{C}^n$. A **basis** of $\mathscr{L}$ is any linearly independent finite set B with $\mathscr{L} = \operatorname{span}(B)$.[3] Note that by Corollary 2.3.5 all bases of $\mathscr{L}$ must have the same cardinality. We call this cardinality the **dimension** of $\mathscr{L}$, and denote it by $\dim(\mathscr{L}) \in [n+1]$. If $\mathscr{L}$ is the subspace containing only the zero vector, we say that $\mathscr{L}$ is the **trivial subspace** and has dimension zero.

Example 2.3.6 (The Dimension of $\mathbb{C}^n$)**.** *The n standard basis vectors $\{\mathbf{e}_j\}_{j\in[n]} \subset \mathbb{C}^n$ are indeed a basis of $\mathbb{C}^n$ because they are linearly independent and satisfy $\mathbb{C}^n = \operatorname{span}\left(\{\mathbf{e}_j\}_{j\in[n]}\right)$. As a result, we can see that the dimension of $\mathbb{C}^n$ is n.*

Exercise 2.3.7. *Let $\mathscr{L}$ be a linear subspace of $\mathbb{C}^n$. Use Lemma 2.3.4 to show that any linearly independent set of vectors $B \subset \mathscr{L}$ has cardinality $\leq n$.*

Exercise 2.3.8. *Let $\mathscr{L} \subset \mathbb{C}^n$ be a linear subspace. Prove that the dimension of $\mathscr{L}$ is at most n.*

Exercise 2.3.9. *Let $\mathscr{L} \subset \mathbb{C}^n$ be a linear subspace. Show that any linearly independent set of vectors $B \subset \mathscr{L}$ has cardinality $\leq$ the dimension of $\mathscr{L}$.*

Exercise 2.3.10. *Let $\mathscr{L}, \mathscr{K} \subset \mathbb{C}^n$ be two linear subspaces of $\mathbb{C}^n$ such that $\mathscr{L} \subset \mathscr{K}$. Prove that $\dim(\mathscr{L}) \leq \dim(\mathscr{K})$.*

The following lemma is crucial in several later arguments.

Lemma 2.3.7. *Let $\mathscr{L}, \mathscr{K} \subset \mathbb{C}^n$ be two linear subspaces of $\mathbb{C}^n$ with $\mathscr{L} \cap \mathscr{K} = \{\mathbf{0}\}$. Then $\mathscr{L} \cup \mathscr{K}$ contains $\dim(\mathscr{L}) + \dim(\mathscr{K})$ linearly independent vectors.*

Proof. Let $r = \dim(\mathscr{L})$ and $B = \{\mathbf{b}_j\}_{j\in[r]} \subset \mathscr{L}$ be a basis of $\mathscr{L}$. Similarly, let $s = \dim(\mathscr{K})$ and $A = \{\mathbf{a}_k\}_{k\in[s]} \subset \mathscr{K}$ be a basis of $\mathscr{K}$. We can see that $B \cup A$ must have cardinality $\dim(\mathscr{L}) + \dim(\mathscr{K})$ since $\mathscr{L} \cap \mathscr{K} = \{\mathbf{0}\}$, and neither B nor A can contain $\mathbf{0}$ (recall Exercise 2.3.4). Hence, we will be finished if we can show that $B \cup A$ is linearly independent.

[3] Note that by our definition of "linear subspace" it's not immediately clear that every linear subspace of $\mathbb{C}^n$ has to have a basis. They do, and you can build a basis for any subspace of $\mathbb{C}^n$ in a finite number of steps using the Gram–Schmidt algorithm (see, e.g, Section 2.4).

Suppose for the sake of contradiction that $B \cup A$ is linearly dependent. Then, there exists a nonzero vector $\boldsymbol{\alpha} \in \mathbb{C}^{r+s}$ such that

$$\begin{aligned}\sum_{j\in[r]} \alpha_j \mathbf{b}_j + \sum_{k\in[s]} \alpha_{k+r}\mathbf{a}_k = \mathbf{0} \quad &\Longleftrightarrow \quad \mathscr{L} \ni \sum_{j\in[r]} \alpha_j \mathbf{b}_j = \sum_{k\in[s]} (-\alpha_{k+r})\mathbf{a}_k \in \mathscr{K} \\ &\Longleftrightarrow \quad \sum_{j\in[r]} \alpha_j \mathbf{b}_j = \mathbf{0} \;\text{ and }\; \sum_{k\in[s]} (-\alpha_{k+r})\mathbf{a}_k = \mathbf{0}\end{aligned}$$

since $\mathscr{L} \cap \mathscr{K} = \{\mathbf{0}\}$. Furthermore, at least one of $\sum_{j\in[r]} \alpha_j \mathbf{b}_j$ or $\sum_{k\in[s]}(-\alpha_{k+r})\mathbf{a}_k$ is a nonzero sum since $\boldsymbol{\alpha} \in \mathbb{C}^{r+s}$ is nonzero. However, we then have a contradiction since both A and B are linearly independent. □

Exercise 2.3.11. *Let $\mathscr{L}, \mathscr{K} \subset \mathbb{C}^n$ be two linear subspaces of $\mathbb{C}^n$ with $\dim(\mathscr{L}) + \dim(\mathscr{K}) > n$. Prove that there exists a nonzero vector $\mathbf{x} \in \mathscr{L} \cap \mathscr{K}$.*

Exercise 2.3.12. *Let $\mathscr{L}, \mathscr{K} \subset \mathbb{C}^n$ be two linear subspaces of $\mathbb{C}^n$ with $\dim(\mathscr{L}) + \dim(\mathscr{K}) > n$. Prove that $\mathscr{L} \cap \mathscr{K}$ is a linear subspace of $\mathbb{C}^n$ with $\dim(\mathscr{L} \cap \mathscr{K}) \geq 1$.*

Given a matrix $A \in \mathbb{C}^{m\times n}$, we define the **rank** of A to be the dimension of its column space $\mathcal{C}(A) \subset \mathbb{C}^m$ (which, as a reminder, is the span of the columns of A).

Exercise 2.3.13. *Show that a rank r matrix $A \in \mathbb{C}^{m\times n}$ has exactly r linearly independent columns.*

Exercise 2.3.14. *Show that the rank of a matrix $A \in \mathbb{C}^{m\times n}$ is always $\leq \min\{m, n\}$.*

We define a set of nonzero vectors $\{\mathbf{v}_j\}_{j\in[m]} \subset \mathbb{C}^n$ to be **mutually orthogonal** (or just **orthogonal**) if, for all $j \neq k$, $\langle \mathbf{v}_j, \mathbf{v}_k \rangle = 0$. We will also say that a set containing a single vector $\{\mathbf{v}\} \subset \mathbb{C}^n$ is **trivially orthogonal** since it contains nothing else for $\mathbf{v}$ to fail to be orthogonal with. The next lemma shows that orthogonal vectors are always linearly independent. Hence, they always form a basis of their span.

Lemma 2.3.8. *An orthogonal set of nonzero vectors is always linearly independent.*

Proof. Let $\{\mathbf{y}_j\}_{j\in[m]} \subset \mathbb{C}^n$ be orthogonal nonzero vectors. Suppose that there exist some $\alpha_0, \ldots, \alpha_{m-1} \in \mathbb{C}$ such that

$$\sum_{j\in[m]} \alpha_j \mathbf{y}_j = \mathbf{0}.$$

Let $k \in [m]$. Since the inner product of the zero vector with any other vector is 0, we can see that

$$0 = \left\langle \mathbf{y}_k, \sum_{j\in[m]} \alpha_j \mathbf{y}_j \right\rangle = \sum_{j\in[m]} \alpha_j \langle \mathbf{y}_k, \mathbf{y}_j \rangle = \alpha_k \langle \mathbf{y}_k, \mathbf{y}_k \rangle = \alpha_k \|\mathbf{y}_k\|_2^2.$$

Recalling the properties of norms, we note that since $\mathbf{y}_k \neq \mathbf{0}$, $\|\mathbf{y}_k\|_2^2 > 0$. Hence, $\alpha_k = 0$ must hold for all $k \in [m]$. □

A set of orthogonal vectors in $\mathbb{C}^n$ that all have norm 1 is called an **orthonormal set**. Note that given a set of orthogonal nonzero vectors, we can normalize each of them by replacing $\mathbf{y}_j$ with each $\frac{\mathbf{y}_j}{\|\mathbf{y}_j\|_2}$. This then guarantees that $\left\|\frac{\mathbf{y}_j}{\|\mathbf{y}_j\|_2}\right\|_2 = \frac{1}{\|\mathbf{y}_j\|_2}\|\mathbf{y}_j\|_2 = 1$. Thus, any orthogonal set of nonzero vectors can be turned into an orthonormal set. If a set of orthonormal vectors span a linear subspace $\mathscr{L} \subset \mathbb{C}^n$, we say that they are an **orthonormal basis for** $\mathscr{L}$.

Exercise 2.3.15. *Show that the standard basis vectors $\{\mathbf{e}_j\}_{j\in[n]} \subset \mathbb{C}^n$ form an orthonormal basis of $\mathbb{C}^n$.*

Orthonormal bases have several nice properties. For example, if we know that a vector $\mathbf{x} \in \mathbb{C}^n$ is in the span of an orthonormal basis $\{\mathbf{v}_j\}_{j\in[m]} \subset \mathbb{C}^n$, then we can find an expansion of $\mathbf{x}$ in terms of $\{\mathbf{v}_j\}_{j\in[m]}$ by noting that

$$\mathbf{x} = \sum_{j\in[m]} \alpha_j \mathbf{v}_j \iff \langle \mathbf{v}_k, \mathbf{x}\rangle = \sum_{j\in[m]} \alpha_j \langle \mathbf{v}_k, \mathbf{v}_j\rangle = \alpha_k \|\mathbf{v}_k\|_2^2 = \alpha_k \ \forall k \in [m]. \tag{2.11}$$

Thus, we can easily recover the coefficients $\alpha_j \in \mathbb{C}$ of the linear combination making up $\mathbf{x}$ by taking the inner product of $\mathbf{x}$ with the orthonormal basis vectors. In addition, these coefficients will also satisfy the famous Pythagorean theorem.

Theorem 2.3.9 (The Pythagorean Theorem). *Suppose $\{\mathbf{v}_j\}_{j\in[m]} = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{m-1}\} \subset \mathbb{C}^n$ is an orthonormal set of vectors. Then*

$$\left\| \sum_{j\in[m]} \alpha_j \mathbf{v}_j \right\|_2^2 = \sum_{j\in[m]} |\alpha_j|^2$$

for all $\alpha_0, \ldots, \alpha_{m-1} \in \mathbb{C}$. Equivalently, for any $\mathbf{x} \in$ span $\left(\{\mathbf{v}_j\}_{j\in[m]}\right)$,

$$\|\mathbf{x}\|_2^2 = \sum_{j\in[m]} |\langle \mathbf{x}, \mathbf{v}_j\rangle|^2$$

Proof. Note that the second equation follows immediately from the first since we have $\mathbf{x} = \sum_{j\in[m]} \langle \mathbf{v}_j, \mathbf{x}\rangle \mathbf{v}_j$. To show the first equation let $\alpha_0, \ldots, \alpha_{m-1} \in \mathbb{C}$. Then,

$$\begin{aligned}\left\| \sum_{j\in[m]} \alpha_j \mathbf{v}_j \right\|_2^2 &= \left\langle \sum_{j\in[m]} \alpha_j \mathbf{v}_j, \sum_{k\in[m]} \alpha_k \mathbf{v}_k \right\rangle = \sum_{j\in[m]} \sum_{k\in[m]} \langle \alpha_j \mathbf{v}_j, \alpha_k \mathbf{v}_k\rangle \\ &= \sum_{j\in[m]} \sum_{k\in[m]} \overline{\alpha_j} \alpha_k \langle \mathbf{v}_j, \mathbf{v}_k\rangle = \sum_{j\in[m]} \overline{\alpha_j} \alpha_j \langle \mathbf{v}_j, \mathbf{v}_j\rangle = \sum_{j\in[m]} |\alpha_j|^2.\end{aligned}$$

□

Having hopefully reminded you why orthonormal bases are so great, we will now discuss how to generate one.

2.4 Orthonormal Bases and the Gram–Schmidt Algorithm

Algorithm 1 is an implementation of the Gram–Schmidt Algorithm which, when given a finite set $S \subset \mathbb{C}^n$ as input, outputs an orthonormal basis of $\mathrm{span}(S)$. Before we analyze this algorithm to see that it works as intended we highly recommend that the reader take a close look at it. Here are some recommended exercises to help you pay close attention to how it works.

Exercise 2.4.1. *Run Algorithm 1 on the set*

$$S = \left\{ \begin{pmatrix} 1 \\ \mathbb{i} \end{pmatrix}, \begin{pmatrix} 2+\mathbb{i} \\ 1 \end{pmatrix} \right\} \subset \mathbb{C}^2$$

by hand. Verify that the basis $B \subset \mathbb{C}^2$ it produces for $\mathrm{span}(\mathrm{S})$ *is indeed an orthonormal basis.*

Exercise 2.4.2. *Run Algorithm 1 on the set*

$$S = \left\{ \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix} \right\} \subset \mathbb{C}^3$$

by hand. Verify that the basis $B \subset \mathbb{C}^3$ it produces for $\mathrm{span}(\mathrm{S})$ *is indeed an orthonormal basis.*

When you are finished inspecting Algorithm 1 come back here and we prove a lemma which takes a step toward showing that the set B Algorithm 1 outputs is *always* an orthonormal basis for the span of the input set S.

Lemma 2.4.1. *The set $B \subset \mathbb{C}^n$ output by Algorithm 1 is always orthonormal.*

Proof. First, we observe from Lines 3 and 8 of Algorithm 1 that each $\mathbf{b}_j \in B$ will have norm 1. Thus, it only remains to show that B is orthogonal. To show orthogonality it suffices to show that the set $\{\mathbf{b}_\ell\}_{\ell=0}^{j} = \{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_j\} \subset B$ is orthogonal for all $j \in [|B|]$. We will proceed by induction on j.

To begin, we note that $\{\mathbf{b}_\ell\}_{\ell=0}^{0} = \{\mathbf{b}_0\}$ when $j = 0$ is trivially orthogonal as a singleton set. Now, as our induction hypothesis, assume that $\{\mathbf{b}_\ell\}_{\ell=0}^{j}$ is orthogonal. To show that $\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}$ must also then be orthogonal it suffices to show that $\langle \mathbf{b}_k, \mathbf{b}_{j+1} \rangle$ will be 0 for all integers $k \in [0, j]$. Referring to Lines 6 and 8 of Algorithm 1, and noting that the vector

Algorithm 1 The Gram–Schmidt Algorithm for Finite Sets

```
 1: Input: A finite set $S \subset \mathbb{C}^n$ with at least one nonzero element.
 2: Output: An orthonormal set $B \subset \mathbb{C}^n$ with $\text{span}(B) = \text{span}(S)$.
    # Initialize $S$ and $B$.
 3: Pick a nonzero $\mathbf{x}_0 \in S$, and set $\mathbf{b}_0 := \mathbf{x}_0/\|\mathbf{x}_0\|_2$ and $B = \{\mathbf{b}_0\}$.
 4: Set $S = S \setminus \{\mathbf{0}, \mathbf{x}_0\}$, and initialize $j = 1$.
 5: while $S \neq \{\}$ do
 6:     Pick $\mathbf{x}_j \in S$, and set $\mathbf{y}_j = \mathbf{x}_j - \sum_{\ell=0}^{j-1} \langle \mathbf{b}_\ell, \mathbf{x}_j \rangle \mathbf{b}_\ell$.
        # If $\mathbf{y}_j = \mathbf{0}$ then $\mathbf{x}_j \in \text{span}(B)$ already, so we'll immediately remove this $\mathbf{x}_j$ from $S$
        # in Line 11 and pick a new one. If $\mathbf{y}_j \neq \mathbf{0}$ then $\mathbf{x}_j \notin \text{span}(B)$, so we will add a new
        # element to $B$ so that $\mathbf{x}_j$ will then belong to its new span.
 7:     if $\mathbf{y}_j \neq \mathbf{0}$ then
 8:         Set $\mathbf{b}_j := \mathbf{y}_j/\|\mathbf{y}_j\|_2$ and let $B = B \cup \{\mathbf{b}_j\}$.
 9:         Set j = j+1.
10:     end if
11:     Set $S = S \setminus \{\mathbf{x}_j\}$.
12: end while
13: Return $B$.
```

permanently selected as $\mathbf{x}_j$ in Line 6 has its $\mathbf{y}_j \neq \mathbf{0}$ for all $j \in [|B|]$, we can see that indeed

$$\begin{aligned}
\langle \mathbf{b}_k, \mathbf{b}_{j+1} \rangle &= \left\langle \mathbf{b}_k,\ \frac{1}{\|\mathbf{y}_{j+1}\|_2} \left(\mathbf{x}_{j+1} - \sum_{\ell=0}^{j} \langle \mathbf{b}_\ell, \mathbf{x}_{j+1} \rangle \mathbf{b}_\ell \right) \right\rangle \\
&= \frac{1}{\|\mathbf{y}_{j+1}\|_2} \left(\langle \mathbf{b}_k, \mathbf{x}_{j+1} \rangle - \sum_{\ell=0}^{j} \langle \mathbf{b}_\ell, \mathbf{x}_{j+1} \rangle \langle \mathbf{b}_k, \mathbf{b}_\ell \rangle \right) \\
&= \frac{1}{\|\mathbf{y}_{j+1}\|_2} \left(\langle \mathbf{b}_k, \mathbf{x}_{j+1} \rangle - \langle \mathbf{b}_k, \mathbf{x}_{j+1} \rangle \right) = 0
\end{aligned}$$

for all $k \in [0, j]$, where we have used the inductive hypothesis that $\{\mathbf{b}_\ell\}_{\ell=0}^{j}$ is orthogonal in the last line. As a result we can see that $\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}$ will also be orthogonal whenever $\{\mathbf{b}_\ell\}_{\ell=0}^{j}$ is for all $j = 0, 1, \ldots, |B| - 2$, finishing our induction argument. □

Lemma 2.4.1 guarantees that the output, $B \subset \mathbb{C}^n$, of Algorithm 1 is always orthonormal, but in order for it to be a basis of span(S) we also need that span(B) = span(S). This is established in our next lemma.

Lemma 2.4.2. *The set $B \subset \mathbb{C}^n$ output by Algorithm 1 always satisfies span(B) = span(S).*

Proof. It suffices to show that $\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j}\right) = \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j}\right)$ for all $j \in [|B|]$ (think about why![4]). We will show this by induction on j. To begin, we note that when $j = 0$ we have $\text{span}\left(\{\mathbf{x}_0\}\right) = \text{span}\left(\{\mathbf{b}_0\}\right)$ since $\mathbf{b}_0$ is a nonzero scalar multiple of $\mathbf{x}_0$ (see Line 3). Now, suppose for the sake of induction that $\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j}\right) = \text{span}\left(\{\mathbf{b}_\ell\}_{l=0}^{j}\right)$. We will prove that then $\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right) = \text{span}\left(\{\mathbf{b}_\ell\}_{l=0}^{j+1}\right)$ must also hold in the usual two steps.

$\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right) \subset \text{span}\left(\{\mathbf{b}_\ell\}_{l=0}^{j+1}\right)$: Let $\mathbf{x} \in \text{span}\left(\{\mathbf{x}_l\}_{l=0}^{j+1}\right)$. Then, we can write

$$\mathbf{x} = \alpha_{j+1}\mathbf{x}_{j+1} + \mathbf{y}$$

where $\alpha_{j+1} \in \mathbb{C}$ and $\mathbf{y} \in \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j}\right) = \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j}\right)$. By Lines 6 and 8 of Algorithm 1 we also have that

$$\mathbf{x}_{j+1} = \|\mathbf{y}_{j+1}\|_2\mathbf{b}_{j+1} + \sum_{\ell=0}^{j}\langle \mathbf{b}_\ell, \mathbf{x}_{j+1}\rangle \mathbf{b}_\ell$$

so $\mathbf{x}_{j+1} \in \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}\right)$. Therefore, $\mathbf{x} \in \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}\right)$.

$\text{span}\left(\{\mathbf{b}_\ell\}_{l=0}^{j+1}\right) \subset \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right)$: Let $\mathbf{z} \in \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}\right)$. Then we can write

$$\mathbf{z} = \beta_{j+1}\mathbf{b}_{j+1} + \mathbf{y}$$

where $\beta_{j+1} \in \mathbb{C}$ and $\mathbf{y} \in \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j}\right) = \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j}\right)$. Again, by Lines 6 and 8 of Algorithm 1 we also have that

$$\mathbf{b}_{j+1} = \frac{1}{\|\mathbf{y}_{j+1}\|_2}\left(\mathbf{x}_{j+1} - \sum_{\ell=0}^{j}\langle \mathbf{b}_\ell, \mathbf{x}_j\rangle \mathbf{b}_\ell\right),$$

where the sum $\sum_{\ell=0}^{j}\langle \mathbf{b}_\ell, \mathbf{x}_j\rangle \mathbf{b}_\ell$ above is in $\text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j}\right) = \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j}\right)$. Thus, $\mathbf{b}_{j+1} \in \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right)$ which in turn implies that $\mathbf{z} \in \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right)$.

Having now shown that both $\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right) \subset \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}\right)$ and $\text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}\right) \subset \text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right)$ hold, we conclude that indeed $\text{span}\left(\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}\right) = \text{span}\left(\{\mathbf{b}_\ell\}_{l=0}^{j+1}\right)$. □

Combining Lemmas 2.4.1 and 2.4.2 we obtain the following theorem guaranteeing that Algorithm 1 always produces an orthonormal basis of the span of its input set, as intended.

[4] Note that multiple candidates may be selected to be $\mathbf{x}_j$ in Line 6 before j is actually incremented in Line 9. All such temporarily-selected $\mathbf{x}_j$ candidates simply overwrite one another. However, only the final vector permanently selected to be $\mathbf{x}_j$ in Line 6 has its $\mathbf{y}_j \neq \mathbf{0}$. All other temporarily-selected $\mathbf{x}_j$ candidates must have $\mathbf{y}_j = 0$ in Line 7 in order to be overwritten, which means that they are already in the span of $\{\mathbf{b}_\ell\}_{\ell=0}^{j-1}$. Thus, only the span of the final permanently selected $\mathbf{x}_j$'s really matters.

Algorithm 2 The Gram–Schmidt Algorithm for Subspaces of $\mathbb{C}^n$

```
1: Input: A nontrivial subspace $\mathscr{L} \subset \mathbb{C}^n$ (i.e., an $\mathscr{L} \neq \{\mathbf{0}\}$).
2: Output: An orthonormal set $B \subset \mathbb{C}^n$ with $\mathrm{span}(B) = \mathscr{L}$.
3: Pick $\mathbf{x} \in \mathscr{L} \setminus \{\mathbf{0}\}$ and initialize $B = \{\mathbf{x}/\|\mathbf{x}\|_2\}$.
4: while $\mathscr{L} \not\subset \mathrm{span}(B)$ do
5:     Pick $\mathbf{x} \in \mathscr{L} \setminus \mathrm{span}(B)$.
6:     Let $\mathbf{y} = \mathbf{x} - \sum_{\mathbf{b} \in B} \langle \mathbf{b}, \mathbf{x} \rangle \mathbf{b}$.
7:     Set $B = B \cup \{\mathbf{y}/\|\mathbf{y}\|_2\}$.
8: end while
9: Return $B$.
```

Theorem 2.4.3. *Algorithm 1 always returns an orthonormal basis B of* $\mathrm{span}(S) \subset \mathbb{C}^n$.

The Gram–Schmidt algorithm is also useful for a lot of other theoretical reasons as well, which I would like to briefly mention here (please indulge me!). For example, based on our definition of what a linear subspace of $\mathbb{C}^n$ is, it's is not immediately clear that every such subspace has to have a basis. This can be established by, e.g., analyzing Algorithm 2 which is a variant of Algorithm 1 (except for subspaces). Please go and look it over.

Looking at Algorithm 2 we can see that a slightly modified version of Lemma 2.4.1 will again guarantee that B will remain orthonormal at all times. The main open question here is therefore whether the "while loop" in Line 4 of Algorithm 2 will ever terminate (subspaces are, after all, infinite sets . . . there are many worst-case $\mathbf{x}$ values to pick from in each iteration!). We need not fear, however. The while loop must terminate after at most n iterations no matter what by the Exchange Lemma (Lemma 2.3.4) exactly because B will always be linearly independent (see, e.g., Exercise 2.3.7). More precisely, it will terminate after $\dim(\mathscr{L}) \leq n$ iterations (see, e.g., Exercise 2.3.9). Failing to do so would generate a contradiction. Formalizing this argument proves the following theorem.

Theorem 2.4.4. *Every nontrivial linear subspace $\mathscr{L} \subset \mathbb{C}^n$ has an orthonormal basis.*

As a final thought regarding Gram–Schmidt algorithm variants, we note that they can also be used to expand an orthonormal basis of a low-dimensional subspace of $\mathbb{C}^n$ into a larger orthonormal basis of all of $\mathbb{C}^n$. This fact comes in handy on many occasions. More precisely, suppose that we have an orthonormal basis B of a subspace $\mathscr{L} \subset \mathbb{C}^n$ with $|B| = \dim(\mathscr{L}) < n$ in our possession. Then, we can use Algorithm 3 to extend it to a larger basis $\tilde{B}$ of $\mathbb{C}^n$ with $B \subset \tilde{B}$. Please go take a look at Algorithm 3, paying special attention to its similarities and differences with Algorithm 2.

Looking at Algorithm 3 we can see that it is effectively a continuation of Algorithm 2. That is, Algorithm 3 effectively picks up where Algorithm 2 leaves off and then continues in the exact same way after substituting $\mathscr{L}$ with $\mathbb{C}^n$ everywhere in its "while loop". As a consequence of this substitution, we can use essentially the same reasoning as above to

Algorithm 3 Gram–Schmidt for Extending an Orthonormal Basis

```
Input: An orthonormal basis $B$ of a subspace $\mathscr{L} \subset \mathbb{C}^n$.
Output: An orthonormal basis $\tilde{B}$ of $\mathbb{C}^n$ with $B \subset \tilde{B}$.
Initialize $\tilde{B} = B$.
while $\mathbb{C}^n \not\subset \operatorname{span}\left(\tilde{B}\right)$ do
    Pick $\mathbf{x} \in \mathbb{C}^n \setminus \operatorname{span}\left(\tilde{B}\right)$.
    Let $\mathbf{y} = \mathbf{x} - \sum_{\mathbf{b} \in \tilde{B}} \langle \mathbf{b}, \mathbf{x} \rangle \mathbf{b}$.
    Set $\tilde{B} = \tilde{B} \cup \{\mathbf{y}/\|\mathbf{y}\|_2\}$.
end while
Return $\tilde{B}$.
```

see that Algorithm 3 will indeed output an orthonormal basis $\tilde{B}$ of $\mathbb{C}^n$. Furthermore, the fact that $B \subset \tilde{B}$ is entirely a result of how $\tilde{B}$ is initialized. Formalizing this line of thought proves the following theorem.

Theorem 2.4.5. *Let $B \subset \mathbb{C}^n$ be an orthonormal basis of a linear subspace $\mathscr{L} \subset \mathbb{C}^n$. Then there exists an orthonormal basis $\tilde{B}$ of $\mathbb{C}^n$ such that $B \subset \tilde{B}$.*

Exercise 2.4.3. *Implement a version of Algorithm 3 in the language of your choice*[5] *and use it to complete the orthonormal set*

$$S = \left\{ \frac{1}{2}\begin{pmatrix} 1 \\ -1 \\ 1 \\ \mathrm{i} \end{pmatrix}, \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ -\mathrm{i} \end{pmatrix} \right\} \subset \mathbb{C}^4$$

to an orthonormal basis of all of $\mathbb{C}^4$. Verify that your resulting orthonormal basis set is indeed orthonormal.

Exercise 2.4.4. *Prove that every set of n orthonormal vectors in $\mathbb{C}^n$ is an orthonormal basis of $\mathbb{C}^n$.*

Exercise 2.4.5. *Let $\mathscr{L} \subset \mathbb{C}^n$ be a linear subspace of dimension $r \leq n$. Prove that every set of r orthonormal vectors in $\mathscr{L}$ is an orthonormal basis of $\mathscr{L}$.*

We will now explore yet another important consequence of the Gram–Schmidt algorithm – the existence of a QR factorization for any matrix $A \in \mathbb{C}^{m \times n}$.

[5] The language of your choice can also be "by hand" (recommended). If you do decide to write computer code, though, feel free to ask A.I. for help.

2.4.1 The QR Decomposition of a Matrix

Let's consider what happens when we apply Algorithm 1 to the columns of a matrix $A \in \mathbb{C}^{m\times n}$ so that it's input is $S = \{A_{:,j}\}_{j\in[n]} \subset \mathbb{C}^m$. Even more specifically, suppose that we run Algorithm 1 with $\mathbf{x}_0 = A_{:,0}$, $\mathbf{x}_1 = A_{:,1}$ (or, more generally, = the first column after $A_{:,0}$ that isn't a multiple of $A_{:,0}$), $\mathbf{x}_2 = A_{:,2}$ (or, more generally, = the first column after $\mathbf{x}_1$ that isn't in the span of $\mathbf{x}_0$ and $\mathbf{x}_1$), etc.. First, we know that Algorithm 1 will output an orthonormal basis $B \subset \mathbb{C}^m$ of the column space, $\mathcal{C}(A)$, of A when its finishes. Second, by the definition of rank we also know that $|B| = \text{rank}(A)$. Denote the rank of A by r, and the elements of B by $\{\mathbf{b}_j\}_{j\in[r]}$.

By our analysis of Algorithm 1 we can further see that $A_{:,0} \in \text{span}(\{\mathbf{b}_0\})$, $A_{:,1} \in \text{span}(\{\mathbf{b}_0, \mathbf{b}_1\})$, etc.. More generally, $A_{:,j} \in \text{span}\left(\{\mathbf{b}_\ell\}_{\ell=0}^{\min\{j,r-1\}}\right)$ for all $j \in [n]$. As a consequence, there exist complex numbers $R_{i,j} \in \mathbb{C}$, with $i, j \in [n]$ and $i \leq j$, such that

$$\begin{aligned}
A_{:,0} &= R_{0,0}\mathbf{b}_0 \\
A_{:,1} &= R_{0,1}\mathbf{b}_0 + R_{1,1}\mathbf{b}_1 \\
&\vdots \\
A_{:,j} &= \sum_{\ell=0}^{\min\{j,r-1\}} R_{\ell,j}\mathbf{b}_\ell \;\; \textit{for all } j \in [n].
\end{aligned}$$

Now, define $Q \in \mathbb{C}^{m\times r}$ to be the matrix with the elements of B as its columns, and $R \in \mathbb{C}^{r\times n}$ to be the upper triangular matrix whose nonzero entries are defined above so that

$$Q = \begin{pmatrix} | & & | \\ \mathbf{b}_0 & \cdots & \mathbf{b}_{r-1} \\ | & & | \end{pmatrix} \text{ and } R = \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,r-1} & \cdots & R_{0,n-1} \\ 0 & R_{1,1} & \cdots & R_{1,r-1} & \cdots & R_{1,n-1} \\ 0 & 0 & \ddots & \vdots & & \vdots \\ \vdots & \vdots & 0 & R_{r-1,r-1} & \cdots & R_{r-1,n-1} \end{pmatrix}.$$

Doing so we can see that

$$\begin{pmatrix} | & | & & | \\ A_{0,:} & A_{1,:} & \cdots & A_{:,n-1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{r-1} \\ | & | & & | \end{pmatrix} \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,r-1} & \cdots \\ 0 & R_{1,1} & \cdots & R_{1,r-1} & \cdots \\ 0 & 0 & \ddots & \vdots & \\ \vdots & \vdots & 0 & R_{r-1,r-1} & \cdots \end{pmatrix}.$$

That is, $A = QR$. This is called a **QR decomposition of** A, and it is very useful computationally since both Q and R have special properties. Namely, Q has orthonormal columns, and R is upper triangular. By formalizing the discussion above one may prove the following theorem.

Theorem 2.4.6 (Every Matrix Has a QR Decomposition)**.** *Let $A \in \mathbb{C}^{m \times n}$ be rank r. Then, there exists a matrix $Q \in \mathbb{C}^{m \times r}$ with orthonormal columns, and an upper triangular matrix $R \in \mathbb{C}^{r \times n}$, so that $A = QR$.*

Example 2.4.7. *The following is an example of a QR decomposition for a rank 2 matrix $A \in \mathbb{C}^{4 \times 4}$.*

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

Note that the matrix Q guaranteed by Theorem 2.4.6 is also clearly rank r since Q has orthonormal columns. In fact, with just a bit more work one can further see that R will always be rank r as well. We will save this final rank analysis for Section 2.7, however. For now, let us turn our attention to some implications of the QR decomposition with regard to low rank matrix compression.

Exercise 2.4.6. *Compute a QR decomposition of the matrix*

$$A = \begin{pmatrix} 1 & 2 \\ \mathbb{i} & 1 \end{pmatrix}.$$

Verify that Q has orthonormal columns.

Exercise 2.4.7. *Compute a QR decomposition of the matrix*

$$A = \begin{pmatrix} 1 & 5 & -1 \\ -1 & 1 & -1 \\ 2 & 1 & 1 \end{pmatrix}.$$

Verify that Q has orthonormal columns.

Some Comments on Computing a QR Decomposition of a Matrix: I hope that this section has begun to convince you that the QR decomposition might be interesting. In fact, we will see going forward that the QR decomposition is also incredibly useful – useful enough that I am pretty certain that anyone reading this sentence will likely compute one at some point (probably using a preexisting software package like – these days – MATLAB, SciPy, LAPACK, or . . . there are many!). When you do compute that QR decomposition it's important to point out that it won't be by (shouldn't be by!) running Algorithm 1 on the columns of the matrix. Theoretically Algorithm 1 is fantastic, but in practice a digital computer will likely turn a straightforward coding of Algorithm 1 into the inaccurate numerical equivalent of a reeking garbage scow (i.e., it'll be numerically unstable). In practice QR decompositions are instead computed using Householder reflections which, if interested, you can read about in standard numerical linear algebra texts such as, e.g., [47, 16].

2.5 Near-Optimal Compression of Low Rank Matrices

In this section we briefly consider the minimum number of complex values we need to store in order to fully represent a rank r matrix $A \in \mathbb{C}^{m\times n}$. Clearly, we can always do it by storing all mn entries in A, but can we do better? The answer is definitely "yes" if the matrix is low rank. To see why, consider a QR decomposition of $A \in \mathbb{C}^{m\times n}$, $A = QR$. Recalling that $Q^{m\times r}$ and $R \in \mathbb{C}^{r\times n}$, we immediately see that in fact we can completely represent A by instead storing the at most $mr + nr = r(m+n)$ entries of Q and R. And if, for example, $n = m$ and $r < n/2$, storing the at most $mr + nr = 2nr < n^2$ entries of Q and R will require less memory than directly storing the $mn = n^2$ entries of A.

In fact, however, we can do even better than this by taking full advantage of the structure that a QR decomposition guarantees us. Since, e.g, R is upper triangular we know that it will always have $\frac{(r-1)r}{2}$ zero entries below its main diagonal in predictable positions. Thus, there is no need to actually store those 0-valued entries of R. As a result, we can see that it really suffices to only store

$$mr + rn - \frac{(r-1)r}{2} = r\left(m + n - \frac{r-1}{2}\right) \tag{2.12}$$

complex numbers in order to fully represent both Q and R, and therefore A. *Note that this reduction in entries can have noticeable space-saving effects, especially when we need to store a large number of very large matrices. Further note that this is exactly the case one is in when, e.g., one wants to store the many large weight matrices needed to fully describe a trained deep neural network (recall Section 1.2.3)!*

Exercise 2.5.1. *Show that an upper triangular matrix $R \in \mathbb{C}^{r\times n}$ with $r \leq n$ will always have $\frac{(r-1)r}{2}$ zero entries below its main diagonal.*

The number of complex entries (2.12) one needs to store in order to represent a QR decomposition as described above is not quite optimal. To see why, we note that the dimension of the manifold of rank r matrices in $\mathbb{C}^{m\times n}$ is $(m+n-r)r$ (see, e.g., [21, Chapter 1]), so one should be able to represent any rank r matrix $A \in \mathbb{C}^{m\times n}$ by storing just $(m+n-r)r$ complex values. This means that storing a QR decomposition of A requires storing $\frac{r^2+r}{2}$ additional complex values beyond the theoretical minimum. As we will see, Gaussian elimination can help us reduce this number of additional values closer to 0. Using this as motivation we will now very briefly summarize Gaussian elimination while simultaneously introducing and reviewing a lot of other very useful notation.

2.5.1 A Very Brief Review of Gaussian Elimination, and Some Useful Notation

First let's recall some notation. As mentioned above, we view vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$ as $n \times 1$ matrices. As a result, the inner product $\langle \mathbf{u}, \mathbf{v}\rangle \in \mathbb{C}$ can be viewed as the matrix product of a $1 \times n$ matrix with an $n \times 1$ matrix,

$$\mathbf{u}^*\mathbf{v} = (\overline{u_0}, \overline{u_1}, \ldots, \overline{u_{n-1}}) \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} = \sum_{j \in [n]} \overline{u_j} v_j = \langle \mathbf{u}, \mathbf{v} \rangle,$$

the result of which is a 1×1 matrix (i.e., the scalar $\langle \mathbf{u}, \mathbf{v} \rangle$). We can similarly define the "outer product of two vectors" in $\mathbb{C}^n$ as the product of an $n \times 1$ matrix with a $1 \times n$ matrix. That is, given $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$, their **outer product** is

$$\mathbf{v}\mathbf{u}^* = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} (\overline{u_0}, \overline{u_1}, \ldots, \overline{u_{n-1}}) = \begin{pmatrix} v_0\overline{u_0} & v_0\overline{u_1} & \ldots & v_0\overline{u_{n-1}} \\ v_1\overline{u_0} & v_1\overline{u_1} & \ldots & v_1\overline{u_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1}\overline{u_0} & v_{n-1}\overline{u_1} & \ldots & v_{n-1}\overline{u_{n-1}} \end{pmatrix} \in \mathbb{C}^{n \times n}.$$

Note that this is an $n \times n$ matrix whose $(j, k)^{\text{th}}$ entry is $v_j\overline{u_k} \in \mathbb{C}$.

Next, the **standard basis of** $\mathbb{C}^{m \times n}$ consists of the mn matrices in $\mathbb{C}^{m \times n}$, denoted by $E^{(j,k)} \in \mathbb{C}^{m \times n}$, whose entries are given by

$$\left(E^{(j,k)}\right)_{\ell,h} = \begin{cases} 1 & \text{if } \ell = j \text{ and } h = k \\ 0 & \text{else} \end{cases} \qquad \text{for all } j, \ell \in [m] \text{ and } k, h \in [n].$$

Note that we also have $E^{(j,k)} = \mathbf{e}_j\mathbf{e}_k^*$. We call these matrices the standard basis for $\mathbb{C}^{m \times n}$ because any matrix $A \in \mathbb{C}^{m \times n}$ can be expressed as the linear combination

$$A = \sum_{j \in [m]} \sum_{k \in [n]} A_{j,k} E^{(j,k)} = \sum_{j \in [m]} \sum_{k \in [n]} A_{j,k} \mathbf{e}_j \mathbf{e}_k^*.$$

Continuing, given a vector $\mathbf{v} \in \mathbb{C}^n$, we denote the diagonal matrix in $\mathbb{C}^{n \times n}$ with $\mathbf{v}$ on its diagonal by $\operatorname{diag}(\mathbf{v}) \in \mathbb{C}^{n \times n}$. Equivalently, $\operatorname{diag}(\mathbf{v})$ is the $n \times n$ matrix with entries given by

$$(\operatorname{diag}(\mathbf{v}))_{j,k} = \begin{cases} v_j & \text{if } j = k \\ 0 & \text{else} \end{cases}.$$

Finally, we will denote the vector of all ones in $\mathbb{C}^n$ by $\mathbf{1} \in \mathbb{C}^n$. The following exercises will help you get more familiar with all of this notation.

Exercise 2.5.2. *Let* $\mathbf{v} \in \mathbb{C}^n$. *Show that* $\operatorname{diag}(\mathbf{v}) = \sum_{j \in [n]} v_j \mathbf{e}_j \mathbf{e}_j^* = \sum_{j \in [n]} v_j E^{(j,j)}$.

Exercise 2.5.3. *Let* $\mathbf{v}, \mathbf{u} \in \mathbb{C}^n$. *Show that* $\operatorname{diag}(\mathbf{v})\mathbf{u} = \operatorname{diag}(\mathbf{u})\mathbf{v} \in \mathbb{C}^n$. *As a consequence, show that* $\operatorname{diag}(\mathbf{1})\mathbf{v} = \operatorname{diag}(\mathbf{v})\mathbf{1} = \mathbf{v}$ *holds for all* $\mathbf{v} \in \mathbb{C}^n$.

We can now see that the $n \times n$ **identity matrix**, denoted by $I_n \in \mathbb{C}^{n\times n}$, can be expressed in several equivalent forms. First, we know that its entries are $(I_n)_{j,k} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{else} \end{cases}$, for all $j, k \in [n]$. As a consequence we can see that

$$\begin{aligned} I_n &= \begin{pmatrix} | & | & & | \\ \mathbf{e}_0 & \mathbf{e}_1 & \cdots & \mathbf{e}_{n-1} \\ | & | & & | \end{pmatrix} = \operatorname{diag}(\mathbf{1}) \\ &= \sum_{j\in[n]} E^{(j,j)} = \sum_{j\in[n]} \mathbf{e}_j \mathbf{e}_j^*. \end{aligned}$$

Additionally, we recall that the **inverse of a matrix** $A \in \mathbb{C}^{n\times n}$, if it exists, is the matrix $A^{-1} \in \mathbb{C}^{n\times n}$ satisfying $AA^{-1} = A^{-1}A = I_n$.

Having equipped ourselves with this new notation, we may now more easily and quickly review Gaussian elimination. In short, **Gaussian elimination** is the process of multiplying three types of invertible elementary matrices against a given matrix $A \in \mathbb{C}^{m\times n}$ in order to, usually, make A sparser (i.e., contain more zero entries). These three types of invertible **elementary matrices** are:

1. Rescaling Matrices: These $m \times m$ matrices multiply a given row and/or column of $A \in \mathbb{C}^{m\times n}$ by a scalar $\alpha \in \mathbb{C}$. We will denote them by

$$M(j, \alpha) := \operatorname{diag}(\mathbf{1} + (\alpha - 1)\mathbf{e}_j) = I_m + (\alpha - 1)\mathbf{e}_j \mathbf{e}_j^*$$

for any given $\alpha \in \mathbb{C}$ and $j \in [m]$. If multiplied against $A \in \mathbb{C}^{m\times n}$ from the left, $M(j, \alpha) \in \mathbb{C}^{m\times m}$ will multiply the j^{th} row of A by α. If multiplied against $A \in \mathbb{C}^{m\times n}$ on the right, $M(j, \alpha) \in \mathbb{C}^{n\times n}$ will multiply the j^{th} column of A by α.

Example 2.5.1. *Let* $M(0, \alpha) = \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{C}^{2\times 2}$*. Then, we can see that both*

$$\begin{aligned} M(0,\alpha)\begin{pmatrix} a & b \\ c & d \end{pmatrix} &= \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \alpha a & \alpha b \\ c & d \end{pmatrix}, \text{ and} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} M(0,\alpha) &= \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha a & b \\ \alpha c & d \end{pmatrix}. \end{aligned}$$

Exercise 2.5.4. *Show that* $(M(j, \alpha))^{-1} = I_m + (1/\alpha - 1)\mathbf{e}_j \mathbf{e}_j^* = M\left(j, \alpha^{-1}\right)$ *for all* $\alpha \neq 0$ *and* $j \in [m]$.

2. Summing Matrices: These $m \times m$ matrices add a multiple of one row/column to another row/column. We will denote them by

$$S(j, k, \alpha) := I_m + \alpha E^{(j,k)} = I_m + \alpha \mathbf{e}_j \mathbf{e}_k^*$$

for any given $\alpha \in \mathbb{C}$ and $j, k \in [m]$ with $j \neq k$. Given $A \in \mathbb{C}^{m \times n}$, the product $S(j,k,\alpha)A$ effectively adds α(row k of A) to row j of A, and then stores the result back in row j. Similarly, if $S(j,k,\alpha) \in \mathbb{C}^{n \times n}$ is multiplied against A from the right it will add α(column j of A) to column k of A, and then store the result back in column k.

Example 2.5.2. *Let* $S(0,1,\alpha) = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \in \mathbb{C}^{2 \times 2}$. *Then, we can see that both*

$$S(0,1,\alpha)\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a+\alpha c & b+\alpha d \\ c & d \end{pmatrix}, \text{ and}$$
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} S(0,1,\alpha) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b+\alpha a \\ c & d+\alpha c \end{pmatrix}.$$

Exercise 2.5.5. *Show that* $(S(j,k,\alpha))^{-1} = I_m - \alpha \mathbf{e}_j \mathbf{e}_k^* = S(j,k,-\alpha)$ *for all* $\alpha \in \mathbb{C}$ *and* $j, k \in [m]$ *with* $j \neq k$.

3. <u>Atomic Permutation Matrices:</u> These $m \times m$ matrices swap two rows/columns of a given matrix. We will denote them by

$$P(j,k) := I_m - \mathbf{e}_j\mathbf{e}_j^* - \mathbf{e}_k\mathbf{e}_k^* + \mathbf{e}_j\mathbf{e}_k^* + \mathbf{e}_k\mathbf{e}_j^*$$

for any given $j, k \in [m]$ with $j \neq k$. If multiplied against $A \in \mathbb{C}^{m \times n}$ from the left, $P(j,k) \in \mathbb{C}^{m \times m}$ will swap the j^{th} and k^{th} rows of A. If multiplied against $A \in \mathbb{C}^{m \times n}$ on the right, $P(j,k) \in \mathbb{C}^{n \times n}$ will swap the j^{th} and k^{th} columns of A.

Example 2.5.3. *Let* $P(0,1) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \in \mathbb{C}^{2 \times 2}$. *Then, we can see that both*

$$P(0,1)\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} c & d \\ a & b \end{pmatrix}, \text{ and}$$
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} P(0,1) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} b & a \\ d & c \end{pmatrix}.$$

Exercise 2.5.6. *Show that* $(P(j,k))^{-1} = P(j,k) \in \mathbb{C}^{m \times m}$ *for all* $j, k \in [m]$ *with* $j \neq k$.

Exercise 2.5.7. *Let* $P = \prod_{\ell=0}^{q-1} P(j_\ell, k_\ell) \in \mathbb{C}^{n \times n}$, *where* $j_\ell, k_\ell \in [n]$ *with* $j_\ell \neq k_\ell$ *for all* $\ell \in [q]$, *be a product of* q *atomic permutation matrices. Show that* $P^{-1} = P^*$.

Having briefly reviewed Gaussian elimination, we will now return to our attempt to use a QR decomposition of a low rank matrix to try to compress it as much as possible. We will now show how Gaussian elimination can be used to help us improve on what we have already achieved above.

Back to Near-Optimal Compression of Low Rank Matrices: Consider a QR decomposition of $A \in \mathbb{C}^{m\times n}$, $A = QR$. Recalling that $R \in \mathbb{C}^{r\times n}$ will be upper triangular, we further note that there will be a permutation matrix $P \in \mathbb{C}^{n\times n}$ so that RP will be both upper triangular *and* have $(RP)_{j,j} \neq 0$ for all $j \in [r]$.[6] In particular, one can see that P can always be represented by a product of at most $r-1$ atomic permutation matrices which encode the process of swapping column 1 of R with the first column, j_1, of R that has $R_{1,j_1} \neq 0$, then swapping column 2 with the first column, j_2, that has $R_{2,j_2} \neq 0$, etc.. As a result, we can see that remembering (i.e., storing) P requires us to remember at most $r-1$ values in $[n]$ (i.e., the columns $j_1, j_2, \dots, j_{r-1} \in [n]$ of R satisfying $j_\ell = \min\{k \in [n] \mid R_{\ell,k} \neq 0\}$).

Using that RP will be both upper triangular and have $(RP)_{j,j} \neq 0$ for all $j \in [r]$, we can now further see that there will exist an invertible matrix $T \in \mathbb{C}^{r\times r}$ consisting of a product of at most $\frac{r^2+r}{2}$ elementary summing and rescaling matrices such that

$$TRP = \left(I_r|\tilde{R}\right) \in \mathbb{C}^{r\times n}. \tag{2.13}$$

That is, we can carry out Gaussian elimination to transform the first r columns of RP into the $r \times r$ identity matrix. Note that $\tilde{R} \in \mathbb{C}^{r\times(n-r)}$ in (2.13). Recalling that our goal is to compactly represent $A \in \mathbb{C}^{m\times n}$, we can now see that

$$A = QR = QT^{-1}TRPP^{-1} = (QT^{-1})\left(I_r|\tilde{R}\right)P^*,$$

where we have used both (2.13) and that $P^{-1} = P^*$ in the final equality.

Letting $\tilde{Q} = QT^{-1} \in \mathbb{C}^{m\times r}$ we can finally see that $A = \tilde{Q}\left(I_r|\tilde{R}\right)P^*$. Thus, to represent $A \in \mathbb{C}^{m\times n}$ we need to store $\tilde{Q} \in \mathbb{C}^{m\times r}$, $\tilde{R} \in \mathbb{C}^{r\times(n-r)}$, and $P \in \mathbb{C}^{n\times n}$. Recalling from above that we can store the permutation matrix P by remembering at most $r-1$ values in $[n]$, we finally see that we can always represent any rank r matrix $A \in \mathbb{C}^{m\times n}$ by storing just $mr + nr - r^2$ complex values (the optimal number!), plus at most $r-1$ additional integers in $[n]$. This is a clear improvement over (2.12).

To conclude, we briefly mention that there are other factors we might want to consider when storing A in a factorized form beyond the total number of entries the factorization requires us to store. For example, we might also want to ensure that both $\tilde{Q} \in \mathbb{C}^{m\times r}$ and $\tilde{R} \in \mathbb{C}^{r\times(n-r)}$ are "well behaved". We will describe in some more detail what "well behaved" might mean in Section 3.1, as well as how one might come up with a good low rank matrix to store in the first place. For a journal article that uses related ideas to those discussed in this section to produce a similar compressed representation of a low rank matrix we refer the interested reader to [12]. After finishing Chapter 2 and 3.1 the attentive reader will know everything they need to know in order to begin digesting its contents.

[6] One can revisit Example 2.4.7 see that we do generally need a permutation matrix for this to be true.

2.6 Set Addition, Orthogonal Projections, and Perpendicular Subspaces

We will now discuss even more of the useful properties possessed by orthonormal bases. The first of these are related to set addition.

Definition 2.6.1 (Set Sums, Subtractions, and Rescalings)**.** *Let S and T be subsets of $\mathbb{C}^n$. We define the* ***(Minkowski) sum*** *of S and T, denoted by $S + T$, to be the set*

$$S + T := \{\mathbf{x} + \mathbf{y} \mid \mathbf{x} \in S, \mathbf{y} \in T\}.$$

Similarly, for $\alpha \in \mathbb{C}$, we define the **set rescaling** *$\alpha S \subset \mathbb{C}^n$ to be $\{\alpha\mathbf{x} \mid \mathbf{x} \in S\}$. We also define the subtraction of two sets to be*

$$S - T := S + (-1)T = \{\mathbf{x} - \mathbf{y} \mid \mathbf{x} \in S, \mathbf{y} \in T\} \subset \mathbb{C}^n.$$

Note that if $\mathbf{0} \in S$, then $T \subset S + T$. Similarly, if $\mathbf{0} \in T$, then $S \subset S + T$. As a result, if $\mathbf{0} \in S \cap T$, then $S \cup T \subset S + T$ (check this!). For similar reasons, the sum of two linear subspaces U and V of $\mathbb{C}^n$ will also always be a larger linear subspace of $\mathbb{C}^n$ containing both U and V (i.e., U and V will be subspaces of $U + V$).

Lemma 2.6.2. *Let $U, V \subset \mathbb{C}^n$ both be linear subspaces of $\mathbb{C}^n$. Then $U + V$ is also a linear subspace of $\mathbb{C}^n$.*

Proof. It suffices to show that $\text{span}(U + V) \subset U + V$ and we'll be finished (why?). We can see that

$$\begin{aligned}
\mathbf{x} \in \text{span}(U+V) &\implies \exists p \in \mathbb{N} \text{ s.t. } \mathbf{x} = \sum_{\ell \in [p]} \beta_\ell \mathbf{x}_\ell \text{ with } \{\beta_\ell\}_{\ell\in[p]} \subset \mathbb{C} \ \& \ \{\mathbf{x}_\ell\}_{\ell\in[p]} \subset U + V \\
&\implies \mathbf{x} = \sum_{\ell \in [p]} \beta_\ell (\mathbf{u}_\ell + \mathbf{v}_\ell) \text{ for } \{\mathbf{u}_\ell\}_{\ell\in[p]} \subset U \ \& \ \{\mathbf{v}_\ell\}_{\ell\in[p]} \subset V \\
&\implies \mathbf{x} = \underbrace{\left(\sum_{\ell\in[p]} \beta_\ell \mathbf{u}_\ell\right)}_{=:\mathbf{u}} + \underbrace{\left(\sum_{\ell\in[p]} \beta_\ell \mathbf{v}_\ell\right)}_{=:\mathbf{v}}.
\end{aligned}$$

We are now finished since, above, $\mathbf{u} \in \text{span}(U) = U$ and $\mathbf{v} \in \text{span}(V) = V$. Hence, $\mathbf{x} \in U + V$. □

Exercise 2.6.1. *Let $U, V \subset \mathbb{C}^n$ both be linear subspaces of $\mathbb{C}^n$. Show that*

$$\max\{\dim(U), \dim(V)\} \leq \dim(U + V) \leq \dim(U) + \dim(V),$$

where $\dim(U) \in [n+1]$ *denotes the dimension of U, etc.. When will* $\max\{\dim(U), \dim(V)\} = \dim(U + V)$*? When will* $\dim(U + V) = \dim(U) + \dim(V)$*?*

Exercise 2.6.2. *Let $A, B \in \mathbb{C}^{m\times n}$. Show that if A has rank r and B has rank s, then $A + B$ has rank at most $r + s$.*

As we shall soon see, the sum of two "orthogonal" linear subspaces of $\mathbb{C}^n$, U and V, will behave much more predictably than the sum of two arbitrary linear subspaces of $\mathbb{C}^n$. In particular, orthonormal bases of each summed subspace U and V can be directly combined to create a new orthonormal basis of $U + V$.

Definition 2.6.3 (Perpendicular Subspaces)**.** *Let U and V be linear subspaces of $\mathbb{C}^n$. We say that U and V are **perpendicular**, or **orthogonal**, if $\langle \mathbf{u}, \mathbf{v}\rangle = 0$ for all $\mathbf{u} \in U$ and $\mathbf{v} \in V$. We will also denote this by writing $U \perp V$.*

Lemma 2.6.4. *Suppose that B_U is an orthonormal basis of a linear subspace $U \subset \mathbb{C}^n$, B_V is an orthonormal basis of a linear subspace $V \subset \mathbb{C}^n$, and $U \perp V$. Then $B_U \cup B_V$ is an orthonormal basis of $U + V$.*

Proof. Since B_U and B_V are orthonormal, every element in $B_U \cup B_V$ has norm 1. Thus, we only need to show that $B_U \cup B_V$ is orthogonal. Let $\mathbf{x}, \mathbf{y} \in B_U \cup B_V$. Since B_U is orthogonal, if $\mathbf{x} \in B_U$ and $\mathbf{y} \in B_U$, then $\langle \mathbf{x}, \mathbf{y}\rangle = 0$. Since B_V is orthogonal, if $\mathbf{x} \in B_V$ and $\mathbf{y} \in B_V$, then $\langle \mathbf{x}, \mathbf{y}\rangle = 0$. Since $U \perp V$, if $\mathbf{x} \in B_U$ and $\mathbf{y} \in B_V$, or $\mathbf{x} \in B_V$ and $\mathbf{y} \in \mathcal{B}_U$, then $\langle \mathbf{x}, \mathbf{y}\rangle = 0$. Hence, $B_U \cup B_V$ is orthogonal.

Now we will show that $B_U \cup B_V$ is a basis of $U + V$. Since $B_U \cup B_V$ is orthonormal, its entries are linearly independent, so it remains to show that $\mathrm{span}(B_U \cup B_V) = U + V$. Let $B_U = \{\mathbf{b}_0, \dots, \mathbf{b}_{r-1}\}$ and $B_V = \{\mathbf{d}_0, \dots, \mathbf{d}_{s-1}\}$. Then, for any vector $\mathbf{x} \in \mathbb{C}^n$, it holds that

$$\begin{aligned}
\mathbf{x} \in U + V &\iff \exists \mathbf{u} \in U, \mathbf{v} \in V \text{ such that } \mathbf{x} = \mathbf{u} + \mathbf{v} \\
&\iff \mathbf{x} = \left(\sum_{j\in[r]} \alpha_j \mathbf{b}_j\right) + \left(\sum_{j\in[s]} \beta_j \mathbf{d}_j\right) \text{ for some } \{\alpha_j\}_{j\in[r]} \cup \{\beta_j\}_{j\in[s]} \subset \mathbb{C} \\
&\iff \mathbf{x} \in \mathrm{span}(B_U \cup B_V).
\end{aligned}$$

Therefore, $U + V = \mathrm{span}(B_U \cup B_V)$. □

Corollary 2.6.5. *If $U, V \subset \mathbb{C}^n$ are linear subspaces of $\mathbb{C}^n$, and $U \perp V$, then $dim(U + V) = dim(U) + dim(V)$.*

Given any linear subspace $U \subset \mathbb{C}^n$, we define

$$U^\perp := \{\mathbf{x} \in \mathbb{C}^n \mid \langle \mathbf{x}, \mathbf{y}\rangle = 0 \ \forall \mathbf{y} \in U\}.$$

In other words, $U^\perp$ is the set of all vectors orthogonal to everything in U. We will next show that $U^\perp$ is also a linear subspace of $\mathbb{C}^n$.

Lemma 2.6.6. *Let $U \subset \mathbb{C}^n$ be a linear subspace of $\mathbb{C}^n$. Then, $U^\perp$ is also a linear subspace of $\mathbb{C}^n$.*

Proof. It suffices to show that $\text{span}(U^\perp) \subset U^\perp$ (why?). Let $\mathbf{x} \in \text{span}(U^\perp)$. Then, $\mathbf{x}$ is a linear combination of elements in $U^\perp$ so that $\exists p \in \mathbb{N}$, $\{\mathbf{x}_\ell\}_{\ell\in[p]} \subset U^\perp$, and $\{\alpha_\ell\}_{\ell\in[p]} \subset \mathbb{C}$ with $\mathbf{x} = \sum_{\ell\in[p]} \alpha_\ell \, \mathbf{x}_\ell$. Now we can see that for every $\mathbf{y} \in U$ we have

$$\langle \mathbf{x}, \mathbf{y} \rangle = \left\langle \sum_{\ell\in[p]} \alpha_\ell \mathbf{x}_\ell, \mathbf{y} \right\rangle = \sum_{\ell\in[p]} \overline{\alpha_\ell} \langle \mathbf{x}_\ell, \mathbf{y} \rangle = 0$$

since $\{\mathbf{x}_\ell\}_{\ell\in[p]} \subset U^\perp$. Hence, $\mathbf{x} \in U^\perp$. □

We are now prepared to define orthogonal projections with respect to a given orthonormal set. Let $U = \{\mathbf{u}_j\}_{j\in[r]}$ be an orthonormal subset of $\mathbb{C}^n$. We define the **orthogonal projection of x onto span(U) in terms of U** to be the function $P_U : \mathbb{C}^n \to \text{span}(U)$ defined by

$$P_U(\mathbf{x}) = \sum_{j\in[r]} \langle \mathbf{u}_j, \mathbf{x} \rangle \mathbf{u}_j$$

for all $\mathbf{x} \in \mathbb{C}^n$. Note that this definition explicitly depends on the orthonormal basis U of $\text{span}(U)$ that we started with. The idea behind projecting onto a linear subspace $\text{span}(U)$, however, is that the projection should return the portion of $\mathbf{x}$ "living inside" the linear subspace $\text{span}(U)$. That is, it's the *span* of U that matters to us, not the set U itself. If, e.g., we pick a new orthonormal set V with the same exact span as U, then it really shouldn't matter whether we project onto $\text{span}(U) = \text{span}(V)$ using U or V. We should get the same answer either way. The next result will help us show that this is indeed the case.

Lemma 2.6.7. *Let $U = \{\mathbf{u}_\ell\}_{\ell\in[r]}$ and $V = \{\mathbf{v}_\ell\}_{\ell\in[r]}$ be two orthonormal bases of the same linear subspace $\mathscr{L} = \text{span}(U) = \text{span}(V) \subset \mathbb{C}^n$. Then,*

$$\langle \mathbf{u}_j, \mathbf{x} \rangle = \sum_{\ell\in[r]} \langle \mathbf{v}_\ell, \mathbf{x} \rangle \langle \mathbf{u}_j, \mathbf{v}_\ell \rangle$$

for all $\mathbf{x} \in \mathbb{C}^n$ and $j \in [r]$.

Proof. Let $\mathbf{x} \in \mathbb{C}^n$. Extend V to an orthonormal basis $\tilde{V}$ of all of $\mathbb{C}^n$ by appealing to Theorem 2.4.5. The orthonormal set $\tilde{V}$ will take the form $\tilde{V} = \{\mathbf{v}_0, \dots, \mathbf{v}_{r-1}, \mathbf{w}_r, \dots, \mathbf{w}_{n-1}\}$ for some $\mathbf{w}_r, \dots, \mathbf{w}_{n-1} \in \mathbb{C}^n$. Since $\tilde{V}$ is an orthonormal basis of $\mathbb{C}^n$ we can write $\mathbf{x}$ as

$$\mathbf{x} = \sum_{\ell\in[r]} \langle \mathbf{v}_\ell, \mathbf{x} \rangle \mathbf{v}_\ell + \sum_{\ell=r}^{n-1} \langle \mathbf{w}_\ell, \mathbf{x} \rangle \mathbf{w}_\ell.$$

Additionally, since each $\mathbf{u}_j \in U$ is in the span of V, we have for all $r \leq \ell \leq n-1$ that

$$\langle \mathbf{u}_j, \mathbf{w}_\ell \rangle = \left\langle \sum_{k\in[r]} \alpha_{j,k} \mathbf{v}_k, \mathbf{w}_\ell \right\rangle = \sum_{k\in[r]} \overline{\alpha_{j,k}} \langle \mathbf{v}_k, \mathbf{w}_\ell \rangle = 0$$

since $\tilde{V}$ is orthogonal. Hence,

$$\begin{aligned}\langle \mathbf{u}_j, \mathbf{x}\rangle &= \left\langle \mathbf{u}_j, \sum_{\ell\in[r]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle \mathbf{v}_\ell + \sum_{\ell=r}^{n-1}\langle \mathbf{w}_\ell, \mathbf{x}\rangle \mathbf{w}_\ell \right\rangle \\ &= \sum_{\ell\in[r]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle\langle \mathbf{u}_j, \mathbf{v}_\ell\rangle + \sum_{\ell=r}^{n-1}\langle \mathbf{w}_\ell, \mathbf{x}\rangle\langle \mathbf{u}_j, \mathbf{w}_\ell\rangle \\ &= \sum_{\ell\in[r]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle\langle \mathbf{u}_j, \mathbf{v}_\ell\rangle.\end{aligned}$$

□

Using this lemma allows us to show that the orthogonal projection $P_U : \mathbb{C}^n \to \mathrm{span}(U)$ only depends on $\mathrm{span}(U)$, and not on the orthonormal set U itself.

Theorem 2.6.8. *Let $U = \{\mathbf{u}_j\}_{j\in[m]}$ and $V = \{\mathbf{v}_\ell\}_{\ell\in[m]}$ be two orthonormal bases of the same linear subspace $\mathscr{L} \subset \mathbb{C}^n$. Then, $P_U = P_V$.*

Proof. Let $\mathbf{x} \in \mathbb{C}^n$. Appealing to Lemma 2.6.7, we have that

$$\begin{aligned}P_U(\mathbf{x}) = \sum_{j\in[m]}\langle \mathbf{u}_j, \mathbf{x}\rangle \mathbf{u}_j &= \sum_{j\in[m]}\left(\sum_{\ell\in[m]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle\langle \mathbf{u}_j, \mathbf{v}_\ell\rangle\right)\mathbf{u}_j \\ &= \sum_{\ell\in[m]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle\left(\sum_{j\in[m]}\langle \mathbf{u}_j, \mathbf{v}_\ell\rangle \mathbf{u}_j\right) = \sum_{\ell\in[m]}\langle \mathbf{v}_\ell, \mathbf{x}\rangle \mathbf{v}_\ell = P_V(\mathbf{x}),\end{aligned}$$

where we have also used that each $\mathbf{v}_\ell \in V$ is in the span of U (recall (2.11)). □

We now know that an orthogonal projection only depends on the linear subspace of $\mathbb{C}^n$ onto which one projects. Thus, for any linear subspace $\mathscr{L} \subset \mathbb{C}^n$ we can define **the orthogonal projection onto** $\mathscr{L}$, denoted by $P_{\mathscr{L}} : \mathbb{C}^n \to \mathscr{L}$, to be $P_{\mathscr{L}} := P_U$ where U is any orthonormal basis of $\mathscr{L}$ you like.

Example 2.6.9. *The orthogonal projection onto the x-axis of $\mathbb{C}^2$ is the function $P_{\mathrm{x-axis}}$ from $\mathbb{C}^2$ to the x-axis which sends the vector $\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \in \mathbb{C}^2$ to the vector*

$$P_{\mathrm{x-axis}}(\mathbf{x}) = \left\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\rangle \begin{pmatrix} 1 \\ 0 \end{pmatrix} = x_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_0 \\ 0 \end{pmatrix}.$$

Exercise 2.6.3. *Let $\mathscr{L}$ be a linear subspace of $\mathbb{C}^n$. Verify that $P_{\mathscr{L}} : \mathbb{C}^n \to \mathscr{L}$ is a linear function (i.e., that $P_{\mathscr{L}}(\alpha\mathbf{x}+\beta\mathbf{y}) = \alpha P_{\mathscr{L}}(\mathbf{x})+\beta P_{\mathscr{L}}(\mathbf{y})$ holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ and $\alpha, \beta \in \mathbb{C}$).*

Exercise 2.6.4. *Let $B = \{\mathbf{b}_j\}_{j\in[r]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathscr{L} = \mathrm{span}(B)$. Complete B to be an orthonormal basis $\tilde{B} = \{\mathbf{b}_j\}_{j\in[r]} \cup \{\mathbf{u}_\ell\}_{\ell=r}^{n-1} \subset \mathbb{C}^n$ of all of $\mathbb{C}^n$ using Theorem 2.4.5. Prove that $\{\mathbf{u}_\ell\}_{\ell=r}^{n-1}$ is an orthonormal basis of $\mathscr{L}^\perp$.*

The following theorem characterizes many of the most important properties of orthogonal projections.

Theorem 2.6.10. *Let $\mathscr{L}$ be a linear subspace of $\mathbb{C}^n$, and let $\mathbf{x} \in \mathbb{C}^n$.*

1. *If $\mathbf{x} \in \mathscr{L}$ then $P_{\mathscr{L}}(\mathbf{x}) = \mathbf{x}$. As a consequence, $P_{\mathscr{L}}\left(P_{\mathscr{L}}(\mathbf{x})\right) = P_{\mathscr{L}}(\mathbf{x})$ always holds.*
2. *$\mathbf{x} - P_{\mathscr{L}}(\mathbf{x}) \in \mathscr{L}^\perp$.*
3. *$\|\mathbf{x}\|_2^2 = \|P_{\mathscr{L}}(x)\|_2^2 + \|\mathbf{x} - P_{\mathscr{L}}(\mathbf{x})\|_2^2$.*

Proof. We prove each part below.

1. See Exercise 2.6.5.
2. Let $U = \{\mathbf{u}_j\}_{j\in[m]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathscr{L}$, and $\mathbf{y} \in \mathscr{L}$. Then $\mathbf{y} = \sum_{j\in[m]} \alpha_j \mathbf{u}_j$, so that

$$\begin{aligned}
\langle \mathbf{x} - P_{\mathscr{L}}(\mathbf{x}), \mathbf{y}\rangle &= \sum_{j\in[m]} \alpha_j \langle \mathbf{x} - P_{\mathscr{L}}(\mathbf{x}), \mathbf{u}_j\rangle \\
&= \sum_{j\in[m]} \alpha_j \left(\langle \mathbf{x}, \mathbf{u}_j\rangle - \langle P_{\mathscr{L}}(\mathbf{x}), \mathbf{u}_j\rangle\right) \\
&= \sum_{j\in[m]} \alpha_j \left(\langle \mathbf{x}, \mathbf{u}_j\rangle - \left\langle \sum_{\ell\in[m]} \langle \mathbf{u}_\ell, \mathbf{x}\rangle \mathbf{u}_\ell, \mathbf{u}_j \right\rangle\right) \\
&= \sum_{j\in[m]} \alpha_j \left(\langle \mathbf{x}, \mathbf{u}_j\rangle - \overline{\langle \mathbf{u}_j, \mathbf{x}\rangle}\right) = 0,
\end{aligned}$$

   since $\langle \mathbf{x}, \mathbf{u}_j\rangle = \overline{\langle \mathbf{u}_j, \mathbf{x}\rangle}$.

3. From (1) and (2) above we know that $P_{\mathscr{L}}(\mathbf{x})$ and $\mathbf{x} - P_{\mathscr{L}}(\mathbf{x})$ are orthogonal. Hence, normalizing them will produce an orthonormal set whose span contains $\mathbf{x}$. This part now follows from the Pythagorean theorem (see Theorem 2.3.9 and Figure 2.2).

□

Theorem 2.6.10 tells us that we can write $\mathbb{C}^n = \mathscr{L} + \mathscr{L}^\perp$ for any linear subspace $\mathscr{L} \subset \mathbb{C}^n$. In some sense we already know this though – recall, e.g., Exercise 2.6.4! The main contribution of Theorem 2.6.10 is that it expresses this fact in a much simple way using orthogonal projections. This more simply expressed property then also allows for a simpler application of the Pythagorean theorem (see Figure 2.2).

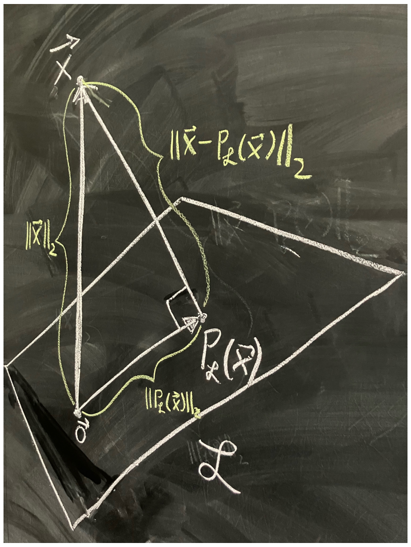
$\vec{x}$
$\|\vec{x}-P_{\mathscr{L}}(\vec{x})\|_2$
$\|\vec{x}\|_2$
$P_{\mathscr{L}}(\vec{x})$
$\|P_{\mathscr{L}}(\vec{x})\|_2$
$\vec{0}$
$\mathscr{L}$

Figure 2.2: A pictorial representation of the projection of $\mathbf{x} \in \mathbb{C}^n$ onto a linear subspace $\mathscr{L} \subset \mathbb{C}^n$. Note that the right triangle whose hypotenuse is of length $\|\mathbf{x}\|_2$ will in fact be entirely contained in the two-dimensional linear subspace spanned by $\{P_{\mathscr{L}}(\mathbf{x}), \mathbf{x} - P_{\mathscr{L}}(\mathbf{x})\}$.

Exercise 2.6.5. *Prove part (1) of Theorem 2.6.10.*

The next lemma demonstrates yet another incredibly useful way of characterizing what the orthogonal projection onto a linear subspace actually does.

Lemma 2.6.11. *Let $\mathbf{x} \in \mathbb{C}^n$. Then $\|\mathbf{x} - P_{\mathscr{L}}(\mathbf{x})\|_2 < \|\mathbf{x} - \mathbf{y}\|_2$ for all $\mathbf{y} \in \mathscr{L} \setminus \{P_{\mathscr{L}}(\mathbf{x})\}$ (i.e, for all $\mathbf{y} \in \mathscr{L}$ with $\mathbf{y} \neq P_{\mathscr{L}}(\mathbf{x})$).*

Proof. We have from Theorem 2.6.10 that

$$\begin{aligned}
\|\mathbf{x}-\mathbf{y}\|_2^2 &= \|P_{\mathscr{L}}(\mathbf{x}-\mathbf{y})\|_2^2 + \|(\mathbf{x}-\mathbf{y}) - P_{\mathscr{L}}(\mathbf{x}-\mathbf{y})\|_2^2 \\
&= \|P_{\mathscr{L}}(\mathbf{x}) - P_{\mathscr{L}}(y)\|_2^2 + \|\mathbf{x} - P_{\mathscr{L}}(\mathbf{x}) - \mathbf{y} + P_{\mathscr{L}}(\mathbf{y})\|_2^2 \\
&= \|P_{\mathscr{L}}(\mathbf{x}) - \mathbf{y}\|_2^2 + \|\mathbf{x} - P_{\mathscr{L}}(\mathbf{x})\|_2^2 \\
&> \|\mathbf{x} - P_{\mathscr{L}}(\mathbf{x})\|_2^2.
\end{aligned}$$

Here we have used that $P_{\mathscr{L}}(\mathbf{y}) = \mathbf{y}$ for all $\mathbf{y} \in \mathscr{L}$ and that $\|P_{\mathscr{L}}(\mathbf{x}) - \mathbf{y}\|_2 > 0$ must hold since $P_{\mathscr{L}}(\mathbf{x}) - \mathbf{y} \neq \mathbf{0}$. □

Looking at Lemma 2.6.11 we can see that $P_{\mathscr{L}}(\mathbf{x}) \in \mathscr{L}$ is the *unique* closest point to $\mathbf{x}$ in $\mathscr{L}$ with respect to ℓ^2-distances (recall Figure 2.2 as well). As a consequence, we can see that in fact

$$P_{\mathscr{L}}(\mathbf{x}) = \arg\min_{\mathbf{y} \in \mathscr{L}} \|\mathbf{x} - \mathbf{y}\|_2$$

also holds.[7] That is, we could have defined $P_{\mathscr{L}}(\mathbf{x})$ to be the closest point in $\mathscr{L}$ to $\mathbf{x}$ in the first place if we had wanted. We will next discuss how to represent $P_{\mathscr{L}}$ as a matrix.

2.6.1 Representing Orthogonal Projections with Matrices

The following fundamental matrices are used to represent all orthogonal projections.

Definition 2.6.12 (Orthonormal and Unitary Matrices)**.** *A matrix $Q \in \mathbb{C}^{m\times n}$ with orthonormal columns will be called an* ***orthonormal matrix****. If $Q \in \mathbb{C}^{n\times n}$ is both orthonormal and square we will call it a* ***unitary matrix****.*

In fact the attentive reader will recognize that we have already been introduced to orthonormal matrices. In particular, the "Q" in a QR decomposition of a given matrix is always an orthonormal matrix. The next few highly recommended exercises will introduce you to some of the very useful properties of orthonormal matrices.

Exercise 2.6.6. *Let $Q \in \mathbb{C}^{m\times n}$ be an orthonormal matrix. Prove that $n \leq m$.*

Exercise 2.6.7. *Let $Q \in \mathbb{C}^{m\times n}$. Show that $Q^*Q = I_n$ if and only if Q is orthonormal.*

Exercise 2.6.8. *Let $Q \in \mathbb{C}^{m\times n}$ be an orthonormal matrix and $\mathbf{x}, \mathbf{y} \in \mathbb{C}^m$. Show that $(I_m - QQ^*)\mathbf{x} = \mathbf{x} - QQ^*\mathbf{x}$ is orthogonal to $QQ^*\mathbf{y}$.*

Let $B = \{\mathbf{q}_\ell\}_{\ell\in[r]} \subset \mathbb{C}^n$ be an orthonormal basis of a linear subspace $\mathscr{L}$. We can form an orthonormal matrix $Q \in \mathbb{C}^{n\times r}$ by letting the columns of Q be the elements of B so that $Q_{:,\ell} = \mathbf{q}_\ell$ for all $\ell \in [r]$, so that

$$Q = \begin{pmatrix} | & | & & | \\ \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{r-1} \\ | & | & & | \end{pmatrix}.$$

We can represent the orthogonal projection onto $\mathscr{L} = \mathcal{C}(Q) = \operatorname{span}(B)$ using an **orthogonal projection matrix** $QQ^* \in \mathbb{C}^{n\times n}$ by

$$P_{\mathscr{L}} = QQ^* = \sum_{j\in[r]} \mathbf{q}_j\mathbf{q}_j^*. \tag{2.14}$$

To see that (2.14) holds it suffices to check that $P_{\mathscr{L}}(\mathbf{x}) = QQ^*\mathbf{x} = \left(\sum_{j\in[r]} \mathbf{q}_j\mathbf{q}_j^*\right)\mathbf{x}$ for all $\mathbf{x} \in \mathbb{C}^n$ (see Exercise 2.6.9). Let $\mathbf{x} \in \mathbb{C}^n$. We have that

[7] Let $S \subset \mathbb{C}^n$ and $f : \mathbb{C}^n \to \mathbb{R}^+$ map $\mathbb{C}^n$ into the nonnegative real numbers. Here and throughout the remainder of the book, "$\arg\min_{\mathbf{y}\in S} f(\mathbf{y})$" returns an element $\mathbf{x} \in S$ satisfying $f(\mathbf{x}) = \min_{\mathbf{y}\in S} f(\mathbf{y})$. If there are many such minimizers it can return any of them (i.e., it might be effectively set-valued). Though it will not be an issue in this book, it's generally important to make sure this notation is well-defined before using it. In particular, one needs to make sure that the minimum of f over S is actually attained by at least one element of S.

$$\begin{aligned} QQ^*\mathbf{x} &= Q\begin{pmatrix} - & \overline{\mathbf{q}_0} & - \\ & \vdots & \\ - & \overline{\mathbf{q}_{r-1}} & - \end{pmatrix}\mathbf{x} = Q\begin{pmatrix} \langle \mathbf{q}_0, \mathbf{x}\rangle \\ \vdots \\ \langle \mathbf{q}_{r-1}, \mathbf{x}\rangle \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{r-1} \\ | & | & & | \end{pmatrix}\begin{pmatrix} \langle \mathbf{q}_0, \mathbf{x}\rangle \\ \vdots \\ \langle \mathbf{q}_{r-1}, \mathbf{x}\rangle \end{pmatrix} \\ &= \sum_{j\in[r]} \mathbf{q}_j \langle \mathbf{q}_j, \mathbf{x}\rangle = P_{\mathscr{L}}(\mathbf{x}) \\ &= \sum_{j\in[r]} \mathbf{q}_j \mathbf{q}_j^* \mathbf{x} = \left(\sum_{j\in[r]} \mathbf{q}_j \mathbf{q}_j^*\right)\mathbf{x}. \end{aligned}$$

Using (2.14) we can also establish the equivalence of orthogonal projection matrices built from orthonormal matrices with the same column spaces.

Lemma 2.6.13. *Let $Q, V \in \mathbb{C}^{m\times n}$ be two orthonormal matrices with the same column span (i.e., with $\mathcal{C}(Q) = \mathcal{C}(V)$). Then $QQ^* = VV^*$.*

Proof. It suffices to show that $QQ^*\mathbf{x} = VV^*\mathbf{x}$ for all $\mathbf{x} \in \mathbb{C}^n$ (see Exercise 2.6.9). Using Theorem 2.6.8 and (2.14) we have that

$$QQ^*\mathbf{x} = P_{\mathcal{C}(Q)}(\mathbf{x}) = P_{\mathcal{C}(V)}(\mathbf{x}) = VV^*\mathbf{x}$$

for all $\mathbf{x} \in \mathbb{C}^n$. We are now finished by Exercise 2.6.9. □

Exercise 2.6.9. *Let $A, B \in \mathbb{C}^{m\times n}$. Suppose that $A\mathbf{x} = B\mathbf{x}$ for all $\mathbf{x} \in \mathbb{C}^n$. Show that $A = B$.*

Exercise 2.6.10. *Let $A, B \in \mathbb{C}^{m\times n}$ and suppose that $S \subset \mathbb{C}^n$ is a basis of $\mathbb{C}^n$. Show that $A = B$ if $A\mathbf{x} = B\mathbf{x}$ holds for all $\mathbf{x} \in S$.*

Exercise 2.6.11. *Show that every orthogonal projection matrix $P_{\mathscr{L}} \in \mathbb{C}^{n\times n}$ satisfies $P_{\mathscr{L}}^* = P_{\mathscr{L}}$.*

The next theorem shows that orthonormal projection matrices built from unitary matrices are always equivalent to the identity matrix.

Theorem 2.6.14. *The following are equivalent:*

1. *$U \in \mathbb{C}^{n\times n}$ is unitary,*
2. *$U^*U = I_n$,*
3. *U^* is unitary, and*
4. *$UU^* = I_n$.*

Proof.

$\underline{(2) \iff (1)}$: Let $U \in \mathbb{C}^{n\times n}$ and set $\mathbf{u}_j := U_{:,j} \in \mathbb{C}^n$ for all $j \in [n]$. Note that $(U^*U)_{\ell,k} = \langle \mathbf{u}_\ell, \mathbf{u}_k \rangle$ for all $\ell, k \in [n]$. As a result we can see that

$$\begin{aligned} U^*U = I_n &\iff (U^*U)_{\ell,k} = (I_n)_{\ell,k} \;\; \forall \ell, k \in [n] \\ &\iff \langle \mathbf{u}_\ell, \mathbf{u}_k \rangle = \begin{cases} 1 & \text{if } \ell = k \\ 0 & \text{else} \end{cases} \\ &\iff \{\mathbf{u}_\ell\}_{\ell \in [n]} \subset \mathbb{C}^n \text{ is an orthonormal set} \\ &\iff U \text{ is unitary.} \end{aligned}$$

Hence, U is unitary if and only if $U^*U = I_n$.

$\underline{(1) \implies (4)}$: Let $U \in \mathbb{C}^{n\times n}$ be unitary. Then $\mathcal{C}(U) = \mathbb{C}^n$ (see Exercise 2.4.4). Since $\mathbb{C}^n = \operatorname{span}\{\mathbf{e}_j\}_{j\in[n]}$ (i.e., $\mathcal{C}(I_n) = \mathbb{C}^n$) Lemma 2.6.13 tells us that $UU^* = I_n I_n^* = I_n$.

$\underline{(4) \iff (3)}$: This is the same as (1) $\iff$ (2) with U replaced by U^*.

$\underline{(3) \implies (2)}$: This is the same as (1) $\implies$ (4) with U replaced by U^*. □

An additional consequence of Theorem 2.6.14 is that a matrix $U \in \mathbb{C}^{n\times n}$ is unitary if and only if $U^* = U^{-1}$. Given this, we can see that we have already met an important family of unitary matrices – the permutation matrices (recall Exercise 2.5.7).

Exercise 2.6.12. *Let $U, V \in \mathbb{C}^{n\times n}$ both be unitary. Show that both $UV \in \mathbb{C}^{n\times n}$ and $VU \in \mathbb{C}^{n\times n}$ are then also unitary.*

Exercise 2.6.13. *Let $U \in \mathbb{C}^{n\times n}$ be unitary. Prove that $\|U\mathbf{x}\|_2 = \|\mathbf{x}\|_2$ for all $\mathbf{x} \in \mathbb{C}^n$.*

Exercise 2.6.14. *Let $B = \{\mathbf{b}_j\}_{j\in[r]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathscr{L} = \operatorname{span}(B)$. Complete B to be an orthonormal basis $\tilde{B} = \{\mathbf{b}_j\}_{j\in[r]} \cup \{\mathbf{u}_\ell\}_{\ell=r}^{n-1} \subset \mathbb{C}^n$ of all of $\mathbb{C}^n$ using Theorem 2.4.5. Let $Q \in \mathbb{C}^{n\times r}$ be the orthonormal matrix with $Q_{:,j} = \mathbf{b}_j$ for all $j \in [r]$ and $U \in \mathbb{C}^{n\times(n-r)}$ be the orthonormal matrix with $U_{:,k} = \mathbf{u}_{r+k}$ for all $k \in [n-r]$. Prove that the orthogonal projection onto $\mathscr{L}^\perp$, $P_{\mathscr{L}^\perp} : \mathbb{C}^n \to \mathscr{L}^\perp$, has the following properties.*

1. $P_{\mathscr{L}^\perp} = UU^*$ (Hint: Recall Exercise 2.6.4.).

2. *Show that $P_{\mathscr{L}}(\mathbf{x}) + P_{\mathscr{L}^\perp}(\mathbf{x}) = \mathbf{x} = I_n\mathbf{x}$ holds for all $\mathbf{x} \in \mathbb{C}^n$. Conclude that $P_{\mathscr{L}^\perp} = I_n - P_{\mathscr{L}}$.*

3. *Show that $UU^* = I_n - QQ^* \in \mathbb{C}^{n\times n}$.*

Lemma 2.6.15. *Let $\mathscr{L}$ and $\mathscr{T}$ be linear subspaces of $\mathbb{C}^n$ such that $\mathscr{L}^\perp = \mathscr{T}$. Then, $\mathscr{T}^\perp = \mathscr{L}$ also holds (i.e., $\left(\mathscr{L}^\perp\right)^\perp = \mathscr{L}$).*

Proof. We must show that both $\mathscr{L} \subset \mathscr{T}^\perp$ and that $\mathscr{T}^\perp \subset \mathscr{L}$ hold.

$\underline{\mathscr{L} \subset \mathscr{T}^\perp}$: Let $\mathbf{x} \in \mathscr{L}$. Then $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ for all $\mathbf{y} \in \mathscr{L}^\perp = \mathscr{T}$ by definition of $\mathscr{L}^\perp$. Hence, $\mathbf{x} \in \mathscr{T}^\perp$.

$\underline{\mathscr{T}^\perp \subset \mathscr{L}}$: Let $\mathbf{x} \in \mathscr{T}^\perp$. By the definition of $\mathscr{T}^\perp$, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ for all $\mathbf{y} \in \mathscr{T} = \mathscr{L}^\perp$. Hence, $P_{\mathscr{L}^\perp}(\mathbf{x}) = \mathbf{0}$. Now we can see that $\mathbf{x} = P_{\mathscr{L}}(\mathbf{x}) + P_{\mathscr{L}^\perp}(\mathbf{x}) = P_{\mathscr{L}}(\mathbf{x})$ (using, e.g., part (2) of Exercise 2.6.14). Thus, $\mathbf{x} \in \mathscr{L}$. □

Now that we have achieved a good understanding of orthogonal projections and orthonormal matrices we are prepared to discuss the least-squares approach to solving systems of linear equations.

2.6.2 Least-Squares Theory for (Approximately) Solving Systems of Linear Equations

Let $A \in \mathbb{C}^{m \times n}$, $\mathbf{b} \in \mathbb{C}^m$, and suppose that we want to solve the equation $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x} \in \mathbb{C}^n$. The least-squares approach aims to do this by minimizing $f(\mathbf{x}) := \|\mathbf{b} - A\mathbf{x}\|_2^2$ as a function of $\mathbf{x} \in \mathbb{C}^n$. To see why this makes sense, observe that $\mathbf{b} \in \mathcal{C}(A) \iff \exists \mathbf{y} \in \mathbb{C}^n$ such that $\mathbf{b} = A\mathbf{y}$ in which case $f(\mathbf{x}) = \|\mathbf{b} - A\mathbf{x}\|_2^2$ will attain its absolute minimum at $f(\mathbf{y}) = 0$. Furthermore, anytime $f(\mathbf{x}) = 0$ it must in fact be the case that $A\mathbf{x} = \mathbf{b}$. Hence, if $A\mathbf{x} = \mathbf{b}$ has solutions we can indeed find one by minimizing f down to 0.

If, on the other hand, $\mathbf{b} \notin \mathcal{C}(A)$ then $A\mathbf{x} = \mathbf{b}$ won't have any solutions and $\inf_{\mathbf{x} \in \mathbb{C}^n} f(\mathbf{x}) = \inf_{\mathbf{x} \in \mathbb{C}^n} \|\mathbf{b} - A\mathbf{x}\|_2^2 > 0$. Nonetheless, there is absolutely nothing stopping us from still minimizing f in hopes of getting "close" to a solution anyways. Observe that by Theorem 2.6.10

$$\begin{aligned}
\|\mathbf{b} - A\mathbf{x}\|_2^2 &= \left\|P_{\mathcal{C}(A)}(\mathbf{b} - A\mathbf{x})\right\|_2^2 + \left\|\mathbf{b} - A\mathbf{x} - P_{\mathcal{C}(A)}(\mathbf{b} - A\mathbf{x})\right\|_2^2 \\
&= \left\|P_{\mathcal{C}(A)}(\mathbf{b}) - A\mathbf{x}\right\|_2^2 + \left\|\mathbf{b} - A\mathbf{x} - P_{\mathcal{C}(A)}(\mathbf{b}) + A\mathbf{x}\right\|_2^2 \\
&= \left\|P_{\mathcal{C}(A)}(\mathbf{b}) - A\mathbf{x}\right\|_2^2 + \left\|\mathbf{b} - P_{\mathcal{C}(A)}(\mathbf{b})\right\|_2^2.
\end{aligned}$$

Above we can see that the first term $\left\|P_{\mathcal{C}(A)}(\mathbf{b}) - A\mathbf{x}\right\|_2^2$ can be minimized to 0 since $P_{\mathcal{C}(A)}(\mathbf{b}) \in \mathcal{C}(A)$, and also that $\left\|\mathbf{b} - P_{\mathcal{C}(A)}(\mathbf{b})\right\|_2^2$ does not depend on $\mathbf{x}$ at all. Hence,

$$\inf_{\mathbf{x} \in \mathbb{C}^n} f(\mathbf{x}) = \inf_{\mathbf{x} \in \mathbb{C}^n} \|\mathbf{b} - A\mathbf{x}\|_2^2 = \left\|\mathbf{b} - P_{\mathcal{C}(A)}(\mathbf{b})\right\|_2^2$$

with the minimum attained when $\mathbf{x}$ satisfies $A\mathbf{x} = P_{\mathcal{C}(A)}(\mathbf{b})$.

The end result of this analysis is that instead of solving $A\mathbf{x} = \mathbf{b}$ we might as well, whenever possible, instead solve $A\mathbf{x} = P_{\mathcal{C}(A)}(\mathbf{b})$ which we know always has a solution.

Algorithm 4 ALGORITHM FOR (APPROXIMATELY) SOLVING $A\mathbf{x} = \mathbf{b}$

```
Input: $A \in \mathbb{C}^{m\times n}$, $\mathbf{b} \in \mathbb{C}^m$.
Output: $\mathbf{x} \in \mathbb{C}^n$ minimizing $f(\mathbf{x}) = \|\mathbf{b} - A\mathbf{x}\|_2^2$.
Compute a QR decomposition of $A$, so that $A = QR$.
Solve $R\mathbf{x} = Q^*\mathbf{b}$ using back substitution.
Return $\mathbf{x}$.
```

Furthermore, we can use a QR decomposition of A to solve $A\mathbf{x} = P_{\mathcal{C}(A)}(\mathbf{b})$ efficiently. Let $A = QR$ be a QR decomposition of A. We have that

$$\begin{aligned} A\mathbf{x} = P_{\mathcal{C}(A)}(\mathbf{b}) &\iff QR\mathbf{x} = P_{C(Q)}(\mathbf{b}) \iff QR\mathbf{x} = QQ^*\mathbf{b} \\ &\iff R\mathbf{x} = Q^*\mathbf{b}. \end{aligned}$$

Furthermore, $R\mathbf{x} = Q^*\mathbf{b}$ can be solved efficiently by back substitution since R is upper triangular. Algorithm 4 outlines how to find the least-squares solution of $A\mathbf{x} = \mathbf{b}$ using a QR decomposition of A.

If $A \in \mathbb{C}^{m\times n}$ is small enough to fit into computer memory and/or accuracy is of principal concern, then one can safely default to directly computing a minimizer of $f(\mathbf{x}) = \|\mathbf{b} - A\mathbf{x}\|_2^2$ using Algorithm 4. If, on the other hand, an approximate least-squares solution suffices and/or A is too large or inaccessible to allow for easy use of Algorithm 4, then one can instead use optimization methods to minimize $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{b} - A\mathbf{x}\|_2^2$ iteratively. In fact, this least-squares problem is important enough that we will discuss it several more times.

Finally, we note that when the rank of $A \in \mathbb{C}^{m\times n}$ is less than n there will be an entire $n - \text{rank}(A)$ dimensional affine subspace of equally good (approximate) solutions to $A\mathbf{x} = \mathbf{b}$. That is, $A(\mathbf{x} + \mathbf{n}) = A\mathbf{x} = \mathbf{b}$ will hold for all $\mathbf{n}$ in the "null space" of A. We will take this as initial motivation to review facts about the null space of a matrix next.

2.7 The Four Fundamental Linear Subspaces of a Matrix

Let $A \in \mathbb{C}^{m\times n}$. The four fundamental linear subspaces of A are:

1. **the column space of** A, $\mathcal{C}(A) = \text{span}\,\{A_{:,j} \mid j \in [n]\} \subset \mathbb{C}^m$,

2. **the null space of** A, or **kernel of** A, $\mathcal{N}(A) = \{\mathbf{x} \in \mathbb{C}^n \mid A\mathbf{x} = \mathbf{0}\} \subset \mathbb{C}^n$,

3. **the column space of** A^*, or **row space of** $\overline{A}$, $\mathcal{C}(A^*) = \text{span}\left\{A^*_{:,j} \mid j \in [m]\right\} \subset \mathbb{C}^n$, and

4. **the null space of** A^*, or **kernel of** A^*, $\mathcal{N}(A^*) = \{\mathbf{y} \in \mathbb{C}^m \mid A^*\mathbf{y} = \mathbf{0}\} \subset \mathbb{C}^m$.

Exercise 2.7.1. *Let $A \in \mathbb{C}^{m\times n}$. Show that the null space of A is a linear subspace of $\mathbb{C}^n$.*

Reviewing facts about each of these linear subspaces, we recall that $r := \text{rank}(A)$ will always equal the dimension of $\mathcal{C}(A)$ by definition. In fact, it also turns out that $A^* \in \mathbb{C}^{n\times m}$ will also always have the same rank as $A \in \mathbb{C}^{m\times n}$.

Theorem 2.7.1. *Let $A \in \mathbb{C}^{m\times n}$. It's always the case that $r = rank(A) = rank(A^*)$.*

Proof. We will use a QR decomposition of A, $A = QR$, with $Q \in \mathbb{C}^{m\times r}$ and $\mathbb{R} \in \mathbb{C}^{r\times n}$. Recall that $\text{rank}(Q) = \text{rank}(A) = r$. Additionally, $\text{rank}(A^*) = \dim(\mathcal{C}(A^*)) = \dim(\mathcal{C}(R^*Q^*))$. Since the columns of Q are orthonormal, so we can extend them to an orthonormal basis B of all of $\mathbb{C}^m$ which takes the form

$$B = \{Q_{:,0}, \dots, Q_{:,r-1}, \mathbf{q}_r, \dots, \mathbf{q}_{m-1}\} \subset \mathbb{C}^m.$$

Now observe that

$$\begin{aligned}
\mathcal{C}(A^*) &= \{A^*\mathbf{y} \mid \mathbf{y} \in \mathbb{C}^m\} = \{R^*Q^*\mathbf{y} \mid \mathbf{y} \in \mathbb{C}^m\} \\
&= \left\{ R^*Q^* \left(\sum_{j\in[r]} \alpha_j Q_{:,j} + \sum_{\ell=r}^{m-1} \beta_\ell \mathbf{q}_\ell \right) \;\middle|\; \{\alpha_j\}_{j\in[r]} \cup \{\beta_\ell\}_{\ell=r}^{m-1} \subset \mathbb{C} \right\} \\
&= \left\{ R^* \left(\sum_{j\in[r]} \alpha_j \mathbf{e}_j \right) \;\middle|\; \{\alpha_j\}_{j\in[r]} \subset \mathbb{C} \right\} \\
&= \mathcal{C}(R^*).
\end{aligned}$$

By the Exchange Lemma (Lemma 2.3.4) it follows that the rank of A^*, which is the size of any basis of $\mathcal{C}(A^*)$, must be less than the number of columns of R^*, which is $r = \text{rank}(A)$. Thus, $\text{rank}(A^*) \leq \text{rank}(A)$. Repeating the argument above with A replaced by A^* similarly shows that $\text{rank}(A) \leq \text{rank}(A^*)$. Combining these two results we learn that $\text{rank}(A) = \text{rank}(A^*)$ must hold. □

Note that the proof of Theorem 2.7.1 above also shows that $\dim(\mathcal{C}(R^*)) = \dim(\mathcal{C}(A^*)) = \text{rank}(A) = r$. Thus, $\text{rank}(R^*)$ equals the number of columns of R^*. Similarly, R is also rank r which equals the number of rows of R. Generally, we will say that any $m \times n$ matrix whose rank matches $\min\{m, n\}$ is **full rank**. Hence, we can see from the argument above that the matrices Q and R resulting from the QR decomposition will always be full rank.

Lemma 2.7.2. *Let $A \in \mathbb{C}^{m\times n}$. Then $\mathcal{N}(A) = \mathcal{C}(A^*)^\perp \subset \mathbb{C}^n$ and $\mathcal{C}(A^*) = \mathcal{N}(A)^\perp \subset \mathbb{C}^n$.*

Proof. By Lemma 2.6.15 it suffices to show that $\mathcal{N}(A) = \mathcal{C}(A^*)^\perp$. Let $\mathbf{x} \in \mathcal{N}(A)$ and consider any given $\mathbf{z} \in \mathcal{C}(A^*)$. By definition, $A\mathbf{x} = \mathbf{0}$ and $\mathbf{z} = A^*\mathbf{y}$ for some $\mathbf{y} \in \mathbb{C}^m$. Hence, we can see that

$$\langle \mathbf{z}, \mathbf{x} \rangle = \langle A^*\mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{y}, A\mathbf{x} \rangle = \langle \mathbf{y}, \mathbf{0} \rangle = 0.$$

Algorithm 5 Algorithm for computing an orthonormal basis of $\mathcal{N}(A)$

```
Input: A rank $r$ matrix $A \in \mathbb{C}^{m \times n}$.
Output: An orthonormal basis of $A$'s null space $\mathcal{N}(A) \subset \mathbb{C}^n$.
Compute a QR decomposition of $A^*$, so that $A^* = QR$. Note that $\mathcal{C}(A^*) = \mathcal{C}(Q)$.
Complete $B = \{Q_{:,j}\}_{j \in [r]}$ to be an orthonormal basis $\tilde{B} = B \cup S$ of all of $\mathbb{C}^n$. The set
  $S$ will be an orthonormal basis of $\mathcal{C}(Q)^\perp = \mathcal{C}(A^*)^\perp = \mathcal{N}(A)$.
Return $S$.
```

Thus, $\mathcal{N}(A) \subset \mathcal{C}(A^*)^\perp$. To see that $\mathcal{C}(A^*)^\perp \subset \mathcal{N}(A)$ also holds, we note that if $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ for all $\mathbf{z} \in \mathcal{C}(A^*)$, then $\langle A^*\mathbf{y}, \mathbf{x} \rangle = 0$ for all $\mathbf{y} \in \mathbb{C}^m$. This in turn implies that $\langle \mathbf{y}, A\mathbf{x} \rangle = 0$ for all $\mathbf{y} \in \mathbb{C}^m$ which means that $\langle A\mathbf{x}, A\mathbf{x} \rangle = \|A\mathbf{x}\|_2^2 = 0$. □

Using Lemma 2.7.2 we can further see that the dimension of $\mathcal{N}(A)$ is $n - r$ since $\mathbb{C}^n = \mathcal{C}(A^*) + \mathcal{C}(A^*)^\perp = \mathcal{C}(A^*) + \mathcal{N}(A)$. Hence, an orthonormal basis of $\mathcal{C}(A^*)$, which will consist of r vectors, can be completed into a larger orthonormal basis of all of $\mathbb{C}^n$ by adding $n - r$ new orthonormal vectors that span $\mathcal{N}(A)$. By encoding this argument as an algorithm we can also create a method for computing an orthonormal basis of the null space of any matrix $A \in \mathbb{C}^{m \times n}$. We can begin by computing an orthonormal basis B of $\mathcal{C}(A^*)$ by, e.g., running Algorithm 1 on the columns of A^*. We can then complete B to an orthonormal basis $\tilde{B} = B \cup S$ of all of $\mathbb{C}^n$ using Algorithm 3. The set S will be an orthonormal basis of $\mathcal{N}(A)$ of size $n - \operatorname{rank}(A)$. See Algorithm 5 for pseudocode.

Exercise 2.7.2. *Let $A \in \mathbb{C}^{m \times n}$ have rank r. Show that $\mathcal{N}(A^*) = \mathcal{C}(A)^\perp \subset \mathbb{C}^m$ and $\mathcal{C}(A) = \mathcal{N}(A^*)^\perp \subset \mathbb{C}^m$. Then, argue that $\dim(\mathcal{N}(A^*)) = m - r$.*

Exercise 2.7.3. *Show that $A \in \mathbb{C}^{n \times n}$ is full rank if and only if $\mathcal{N}(A) = \{\mathbf{0}\}$. Such square matrices are also said to be* **invertible**.

The next lemma will be important soon in Section 3.1. We will prove it here since it depends crucially on our recent revelations regarding null spaces.

Lemma 2.7.3. *Let $A \in \mathbb{C}^{m \times n}$. Then $\mathcal{C}(A^*A) = \mathcal{C}(A^*)$.*

Proof. First we note that

$$\mathcal{C}(A^*A) = \{A^*A\mathbf{y} \mid \mathbf{y} \in \mathbb{C}^n\} = \{A^*\mathbf{z} \mid \mathbf{z} \in \mathcal{C}(A)\} = \left\{A^*P_{\mathcal{C}(A)}\mathbf{x} \mid \mathbf{x} \in \mathbb{C}^m\right\}.$$

Now we can re-express $\mathcal{C}(A^*)$ using that $\mathcal{C}(A)^\perp = \mathcal{N}(A^*) \subset \mathbb{C}^m$ to see that

$$\begin{aligned}\mathcal{C}(A^*) &= \left\{A^*\left(P_{\mathcal{C}(A)}\mathbf{x} + P_{\mathcal{C}(A)^\perp}\mathbf{x}\right) \;\middle|\; \mathbf{x} \in \mathbb{C}^m\right\} = \left\{A^*P_{\mathcal{C}(A)}\mathbf{x} + A^*P_{\mathcal{N}(A^*)}\mathbf{x} \;\middle|\; \mathbf{x} \in \mathbb{C}^m\right\} \\ &= \left\{A^*P_{\mathcal{C}(A)}\mathbf{x} \;\middle|\; \mathbf{x} \in \mathbb{C}^m\right\} = \mathcal{C}(A^*A).\end{aligned}$$

□

Exercise 2.7.4. *Let $A \in \mathbb{C}^{m\times n}$. Prove that $\mathcal{C}(AA^*) = \mathcal{C}(A)$.*

As a consequence of the above, we can see that

$$\operatorname{rank}(A^*A) = \operatorname{rank}(A^*) = \operatorname{rank}(A) = \operatorname{rank}(AA^*).$$

2.8 The Spectral Theorem for Hermitian Matrices

We will now briefly concentrate on a very special type of square matrix which will serve as our doorway to the almighty singular value decomposition in Section 3.1.

Definition 2.8.1. *A matrix $A \in \mathbb{C}^{n\times n}$ is called **Hermitian** if $A = A^*$.*

Exercise 2.8.1. *Let $A \in \mathbb{C}^{m\times n}$. Show that both $AA^* \in \mathbb{C}^{m\times m}$ and $A^*A \in \mathbb{C}^{n\times n}$ are Hermitian.*

Exercise 2.8.2. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. Show that all entries on A's diagonal are real numbers.*

Exercise 2.8.3. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. Show that $\mathcal{N}(A) = \mathcal{C}(A)^\perp \subset \mathbb{C}^n$ and $\mathcal{C}(A) = \mathcal{N}(A)^\perp \subset \mathbb{C}^n$.*

The eigenvalues and eigenvectors of Hermitian matrices have a lot of special properties that we will need later. We will discuss these properties next.

Definition 2.8.2. *An **eigenvalue-eigenvector pair**, or **eigenpair**, of a matrix $A \in \mathbb{C}^{n\times n}$ is a pair $(\lambda, \mathbf{v}) \in \mathbb{C} \times \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that $\mathbf{v} \neq \mathbf{0}$ satisfies $A\mathbf{v} = \lambda\mathbf{v}$.*

Lemma 2.8.3. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. Then all eigenvalues of A are real numbers.*

Proof. Let $(\lambda, \mathbf{v})$ be an eigenpair of A. If $\lambda = 0 \in \mathbb{R}$ we are done. Thus, suppose that $\lambda \neq 0$. Then we have that

$$\begin{aligned}\|\mathbf{v}\|_2^2 &= \langle \mathbf{v}, \mathbf{v}\rangle = \left\langle \frac{1}{\lambda}A\mathbf{v}, \mathbf{v}\right\rangle = \left(1/\overline{\lambda}\right)\langle \mathbf{v}, A^*\mathbf{v}\rangle = \left(1/\overline{\lambda}\right)\langle \mathbf{v}, A\mathbf{v}\rangle = \left(1/\overline{\lambda}\right)\langle \mathbf{v}, \lambda\mathbf{v}\rangle \\ &= \left(\lambda/\overline{\lambda}\right)\|\mathbf{v}\|_2^2.\end{aligned}$$

Since $\mathbf{v}$ is nonzero we know $\|\mathbf{v}\|_2 \neq 0$ so that $\lambda = \overline{\lambda}$ must hold. Hence, $\lambda \in \mathbb{R}$. □

Note that every fixed eigenvalue $\lambda \in \mathbb{C}$ of $A \in \mathbb{C}^{n\times n}$ has an infinite number of associated eigenvectors. In fact, one can see that the set of all eigenvectors corresponding to λ (after adding in the zero vector) is closed under both addition and scalar multiplication so that it forms a linear subspace of $\mathbb{C}^n$. And, this subspace of $\mathbb{C}^n$ is exactly equal to the nullspace of $A - \lambda I_n \in \mathbb{C}^{n\times n}$,

$$\mathcal{N}(A - \lambda I_n) = \left\{\mathbf{v} \in \mathbb{C}^n \;\middle|\; (\lambda, \mathbf{v}) \text{ is an eigenpair of } A\right\} \cup \{\mathbf{0}\}.$$

For this reason we will refer to $\mathcal{N}(A - \lambda I_n)$ as the **eigenspace associated with** λ. Furthermore, we will let an orthonormal basis of this linear subspace be denoted by $B_\lambda \subset \mathbb{C}^n$ for each eigenvalue λ.

Example 2.8.4. *Let $U \in \mathbb{C}^{n \times n}$ be unitary. Then $UU^* = I_n$ so that UU^* has only one nontrivial eigenspace $\mathcal{N}(UU^* - I_n) = \mathbb{C}^n$ associated with its single eigenvalue $\lambda = 1$. Furthermore, its orthonormal basis B_1 will be an orthonormal basis of all of $\mathbb{C}^n$.*

Exercise 2.8.4. *Prove that every matrix $A \in \mathbb{C}^{n \times n}$ with rank $< n$ has at least one nontrivial eigenspace. What is it?*

Exercise 2.8.5. *Prove that $A \in \mathbb{C}^{n \times n}$ has exactly one eigenvalue if and only if it's a scalar multiple of the identity matrix I_n.*

Another important property of Hermitian matrices is that all of their distinct eigenspaces must be orthogonal to one another. This fact is proven in the next lemma.

Lemma 2.8.5. *Let $(\lambda, \mathbf{v})$ and $(\mu, \mathbf{u})$ be two eigenpairs of a Hermitian matrix $A \in \mathbb{C}^{n \times n}$ with $\lambda \neq \mu$. Then $\langle \mathbf{v}, \mathbf{u} \rangle = 0$. As a consequence, $\mathcal{N}(A - \lambda I_n) \perp \mathcal{N}(A - \mu I_n)$.*

Proof. Since $\lambda, \mu \in \mathbb{R}$ are distinct, at least one is nonzero. Without loss of generality let λ be nonzero. Then, $\mu \neq \lambda \implies \frac{\mu}{\lambda} \neq 1 \implies 1 - \frac{\mu}{\lambda} \neq 0$. Since $\lambda \in \mathbb{R} \setminus \{0\}$ we can also see that

$$\langle \mathbf{v}, \mathbf{u} \rangle = \frac{1}{\lambda} \langle A\mathbf{v}, \mathbf{u} \rangle \; = \; \frac{1}{\lambda} \langle \mathbf{v}, A^* \mathbf{u} \rangle \; = \; \frac{1}{\lambda} \langle \mathbf{v}, A\mathbf{u} \rangle \; = \; \frac{1}{\lambda} \langle \mathbf{v}, \mu \mathbf{u} \rangle \; = \; \frac{\mu}{\lambda} \langle \mathbf{v}, \mathbf{u} \rangle.$$

Thus,

$$\left(1 - \frac{\mu}{\lambda}\right) \langle \mathbf{v}, \mathbf{u} \rangle = 0.$$

Hence, it must be the case that $\langle \mathbf{v}, \mathbf{u} \rangle = 0$ since $1 - \frac{\mu}{\lambda} \neq 0$. □

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix whose eigenvalues are $\lambda_0, \dots, \lambda_{m-1} \in \mathbb{R}$. Lemma 2.8.5 implies that the eigenspaces of A will all be orthogonal to one another. As a result, if we let B_{λ_j} be an orthonormal basis for each eigenspace $\mathcal{N}\left(A - \lambda_j I_n\right)$ of A, then

$$B := \bigcup_{j \in [m]} B_{\lambda_j} \subset \mathbb{C}^n \tag{2.15}$$

will be an orthonormal set. In fact, it will also always be the case that B is an orthonormal basis for all of $\mathbb{C}^n$ (we will not prove this here – see, e.g., [25, Chapter 2] or [20, Chapter 14] for corroborating evidence).

Fact 2.8.6. *If $A \in \mathbb{C}^{n\times n}$ is Hermitian then there exists an orthonormal basis of all of $\mathbb{C}^n$ consisting of eigenvectors of A. In particular, the set B in* (2.15) *will be an orthonormal basis of $\mathbb{C}^n$.*

Let $A \in \mathbb{C}^{n\times n}$ be Hermitian and $B = \{\mathbf{b}_j\}_{j\in[n]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of A as defined in (2.15). Form a unitary matrix $U \in \mathbb{C}^{n\times n}$ that contains the elements of B as its columns (i.e., so that $U_{:,j} = \mathbf{b}_j$ for all $j \in [n]$). By the definition of eigenpairs we can see that

$$\begin{aligned} AU &= A\begin{pmatrix} | & | & & | \\ \mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ A\mathbf{b}_0 & A\mathbf{b}_1 & \cdots & A\mathbf{b}_{n-1} \\ | & | & & | \end{pmatrix} \\ &= \begin{pmatrix} | & | & & | \\ \lambda_0\mathbf{b}_0 & \lambda_1\mathbf{b}_1 & \cdots & \lambda_{n-1}\mathbf{b}_{n-1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} \\ | & | & & | \end{pmatrix} \mathrm{diag}(\lambda_0, \ldots, \lambda_{n-1}) \\ &= U\mathrm{diag}(\lambda_0, \ldots, \lambda_{n-1}). \end{aligned}$$

where $\lambda_j \in \mathbb{R}$ refers to the eigenvalue corresponding to $\mathbf{b}_j \in B$. Finally, recalling that U is unitary we can see that multiplying both sides of the equation just above on the right by U^* yields

$$A \;=\; AUU^* \;=\; U\mathrm{diag}(\lambda_0, \ldots, \lambda_{n-1})U^*.$$

This computation together with Lemma 2.8.3, Lemma 2.8.5, and Fact 2.8.6 prove the following theorem (see also, e.g., Theorem 2.5.6 in [25]).

Theorem 2.8.7 (The Full Spectral Decomposition of a Hermitian Matrix)**.** *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. Then there exist $\lambda_0, \ldots, \lambda_{n-1} \in \mathbb{R}$ and a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that*

$$A = U\mathrm{diag}(\lambda_0, \ldots, \lambda_{n-1})U^*.$$

Exercise 2.8.6. *Let $A \in \mathbb{C}^{m\times n}$. Show that all the eigenvalues of the Hermitian matrices $A^*A \in \mathbb{C}^{n\times n}$ and $AA^* \in \mathbb{C}^{m\times m}$ are nonnegative real numbers.*

Theorem 2.8.7 is great, but we'd also like a version that allows us to store low-rank matrices in a compressed form. Let's think about how to develop such a variant – it'll also be good practice for Section 3.1. Recall from our definition of atomic permutation matrices $P(j,k) \in \mathbb{C}^{n\times n}$ (see Example 2.5.3 and the surrounding text) that $P(j,k)$ swaps the j^{th} and k^{th} rows of $A \in \mathbb{C}^{n\times n}$ when multiplied against it on the left, and swaps the j^{th} and k^{th} columns of $A \in \mathbb{C}^{n\times n}$ when multiplied against it on the right. Furthermore, every atomic permutation matrix $P(j,k) \in \mathbb{C}^{n\times n}$ is unitary, as are all products of atomic permutation matrices (see Exercise 2.5.7 and Theorem 2.6.14). Having refamiliarised ourselves with atomic permutation matrices, note that if $P(j,k)$ is applied to both sides of a diagonal matrix simultaneously it will swap its j^{th} and k^{th} diagonal entries. That is,

$$\begin{aligned}&P(j,k)\ \operatorname{diag}(\lambda_0,\dots,\lambda_{j-1},\lambda_j,\lambda_{j+1},\dots,\lambda_{k-1},\lambda_k,\lambda_{k+1},\dots,\lambda_{n-1})\ P(j,k)\\ &\qquad = \ \operatorname{diag}(\lambda_0,\dots,\lambda_{j-1},\lambda_k,\lambda_{j+1},\dots,\lambda_{k-1},\lambda_j,\lambda_{k+1},\dots,\lambda_{n-1}).\end{aligned}$$

Example 2.8.8. *Let* $P(0,2)=\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\in\mathbb{C}^{3\times3}$. *We can see that*

$$\begin{aligned}P(0,2)\ \operatorname{diag}(a,b,c)\ P(0,2) &= \begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\begin{pmatrix}a&0&0\\0&b&0\\0&0&c\end{pmatrix}\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\\ &= \begin{pmatrix}0&0&c\\0&b&0\\a&0&0\end{pmatrix}\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix} = \begin{pmatrix}c&0&0\\0&b&0\\0&0&a\end{pmatrix}\\ &= \operatorname{diag}(c,b,a).\end{aligned}$$

Using these facts about atomic permutation matrices together with Theorem 2.8.7, we can see that there exists a permutation matrix $P=\prod_{\ell\in[q]}P(j_\ell,k_\ell)$ consisting of a product of $q\in\mathbb{N}$ atomic permutation matrices such that

$$\begin{aligned}A = \ &U\ \operatorname{diag}(\lambda_0,\dots,\lambda_{n-1})\ U^* \ = \ U(PP^*)\ \operatorname{diag}(\lambda_0,\dots,\lambda_{n-1})\ (PP^*)U^*\\ &= (UP)(P\ \operatorname{diag}(\lambda_0,\dots,\lambda_{n-1})\ P)(P^*U^*)\\ &= (UP)\ \operatorname{diag}(\tilde\lambda_0,\dots,\tilde\lambda_{n-1})\ (UP)^*,\end{aligned}$$

where $\tilde\lambda_0,\dots,\tilde\lambda_{n-1}$ is a permutation of $\lambda_0,\dots,\lambda_{n-1}\in\mathbb{R}$ satisfying

$$|\tilde\lambda_0|\geq|\tilde\lambda_1|\geq\cdots\geq|\tilde\lambda_{n-1}|.$$

Let $\tilde U=UP$, and note that $\tilde U$ is still a unitary matrix (see, e.g., Exercise 2.6.12).

Continuing, now consider the case where A is not full rank so that $|\tilde\lambda_{n-1}|=0$. In this case we can further compress our spectral decomposition of A using block matrix representations. To begin, let's re-express $\tilde U$ in block form by

$$\tilde U=\begin{pmatrix}V & \tilde U_{:,n-1}\end{pmatrix}\in\mathbb{C}^{n\times n}$$

where $V\in\mathbb{C}^{n\times(n-1)}$ is the orthonormal matrix formed by the first $n-1$ columns of $\tilde U$. Further, let's represent $\operatorname{diag}(\tilde\lambda_0,\dots,\tilde\lambda_{n-1})$ in block form as well by

$$\operatorname{diag}(\tilde\lambda_0,\dots,\tilde\lambda_{n-1})=\begin{pmatrix}D&\mathbf{0}\\\mathbf{0}^*&0\end{pmatrix}\in\mathbb{R}^{n\times n}$$

where $D = \operatorname{diag}(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_{n-2}) \in \mathbb{R}^{(n-1)\times(n-1)}$ and $\mathbf{0}$ is a suitably tall vector of zeroes. Then, we have that

$$A = \begin{pmatrix} V & \tilde{U}_{:,n-1} \end{pmatrix} \begin{pmatrix} D & \mathbf{0} \\ \mathbf{0}^* & 0 \end{pmatrix} \begin{pmatrix} V^* \\ \left(\tilde{U}_{:,n-1}\right)^* \end{pmatrix} = \begin{pmatrix} V & \tilde{U}_{:,n-1} \end{pmatrix} \begin{pmatrix} DV^* \\ \mathbf{0}^* \end{pmatrix} = VDV^*.$$

Note that $V \in \mathbb{C}^{n\times(n-1)}$ above is no longer unitary since it isn't square, but it is still an orthonormal matrix, and D is still a diagonal matrix of real numbers. And, of course, we can repeat this process again if $\tilde{\lambda}_{n-2} = 0$ too, and so on, until we run out of 0 eigenvalues. When will that happen? Well, denote the rank of our Hermitian $A \in \mathbb{C}^{n\times n}$ by $r < n$. The eigenspace associated with the 0 eigenvalue of A is exactly the null space of A so that the orthonormal set B_0 in (2.15) will have $|B_0| = \dim(\mathcal{N}(\mathcal{A})) = n - r$. Hence, we carry out this process $n - r$ total times for all of $\tilde{\lambda}_{n-1} = \tilde{\lambda}_{n-2} = \cdots = \tilde{\lambda}_r = 0$. Formalizing this discussion gives us the following result.

Corollary 2.8.9 (The Compact Spectral Decomposition of a Hermitian Matrix)**.** *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian with rank $r < n$. Then, there exists an orthonormal matrix $U \in \mathbb{C}^{n\times r}$, and $\lambda_0, \ldots, \lambda_{r-1} \in \mathbb{R}$ satisfying*

$$|\lambda_0| \geq |\lambda_1| \geq \cdots |\lambda_{r-1}| > 0,$$

such that

$$A = U \ \operatorname{diag}(\lambda_0, \ldots, \lambda_{r-1}) \ U^*.$$

We end our discussion of the spectral theorem here by noting that Theorem 2.8.7 and Corollary 2.8.9 are really fantastic! They decompose every Hermitian matrix into a product of extremely well behaved (e.g., easily invertible in the full rank case) matrices. Given how much we have used the QR decomposition in this chapter, we hope that the reader can now instinctively anticipate the potential utility of yet another decomposition that in many ways is even nicer (let's be honest – the R in the QR decomposition is just not as nice as the diagonal/unitary combination Theorem 2.8.7 effectively replaces it with). Theorem 2.8.7 and Corollary 2.8.9 do have one major flaw, however. They only apply to one very special type of square matrix! In the next chapter we will remove this flaw by developing a generalization of these Hermitian matrix decompositions that applies to all (even rectangular) matrices.

2.9 Positive (Semi)Definite Matrices

We will now briefly discuss a special class of Hermitian (or symmetric, if real-valued) matrices that are important in optimization, statistics, and applied mathematics more generally. All of the definitions below depend on the following crucial fact which we encourage you to verify.

Exercise 2.9.1. *Let $A \in \mathbb{C}^{n\times n}$ be a Hermitian matrix. Then, $\mathbf{x}^*A\mathbf{x} \in \mathbb{R}\ \forall \mathbf{x} \in \mathbb{C}^n$.*

Definition 2.9.1 (Positive and Negative Definite Matrices)**.** *A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ is* **positive definite** *if $\mathbf{x}^*A\mathbf{x} > 0$ holds $\forall \mathbf{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$. A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ is* **negative definite** *if $\mathbf{x}^*A\mathbf{x} < 0$ holds $\forall \mathbf{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$.*

Definition 2.9.2 (Positive SemiDefinite (PSD) and Negative Semidefinite Matrices)**.** *A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ is* **positive semidefinite** *if $\mathbf{x}^*A\mathbf{x} \geq 0$ holds $\forall \mathbf{x} \in \mathbb{C}^n$. A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ is* **negative semidefinite** *if $\mathbf{x}^*A\mathbf{x} \leq 0$ holds $\forall \mathbf{x} \in \mathbb{C}^n$.*

We encourage the serious reader to do the following exercises in order to help themselves absorb the definitions above.

Exercise 2.9.2. *Let $A \in \mathbb{C}^{n\times n}$. Show that A is negative definite if and only if $-A$ is positive definite.*

Exercise 2.9.3. *Let $A \in \mathbb{C}^{n\times n}$. Prove that both AA^* and A^*A are PSD.*

Exercise 2.9.4. *Let $r \in \mathbb{R}^+$ and $A \in \mathbb{C}^{n\times n}$ be PSD. Show that rA is also PSD.*

Exercise 2.9.5. *Let $A, B \in \mathbb{C}^{n\times n}$ both be PSD. Show that $A + B$ is also PSD.*

The following lemma will be useful below.

Lemma 2.9.3. *Suppose that a Hermitian matrix $A \in \mathbb{C}^{n\times n}$ only has nonzero eigenvalues. Then, A is full rank.*

Proof. We will prove the contrapositive of the desired result after recalling that A is full rank if and only if $\dim(\mathcal{C}(A)) = n$, which in turn will hold if and only if all n columns of A are linearly independent. Now suppose that A is not full rank. Then $\exists \boldsymbol{\alpha} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that $A\boldsymbol{\alpha} = \sum_{j\in[n]} \alpha_j A_{:,j} = \mathbf{0}$. Hence, 0 is an eigenvalue of A with eigvector $\boldsymbol{\alpha}$. □

The discerning reader will see that Lemma 2.9.3 immediately implies that all positive (and negative) definite matrices are full rank and, therefore, invertible (why?). In fact we can say a bit more.

Theorem 2.9.4. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. The following are equivalent:*

1. *A is positive definite.*

2. *All the eigenvalues of A are positive real numbers.*

3. *There exists an invertible matrix $B \in \mathbb{C}^{n\times n}$ such that $A = BB^*$.*

Proof. $\underline{(1) \implies (2)}$: Let λ be an eigenvalue of A with an associated eigenvector $\mathbf{v} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$. We know that λ is a real number by Lemma 2.8.3. Furthermore, since A is positive definite we have that

$$0 < \mathbf{v}^* A \mathbf{v} = \mathbf{v}^*(\lambda \mathbf{v}) = \lambda \underbrace{\|\mathbf{v}\|_2^2}_{>0} \implies \lambda > 0.$$

$\underline{(2) \implies (3)}$: By Theorem 2.8.7 $A = U\mathrm{diag}(\lambda_0, \ldots, \lambda_{n-1})U^*$ where $U \in \mathbb{C}^{n \times n}$ is unitary and the λ_j are the positive real eigenvalues (by assumption) of A. Because the λ_j are nonnegative we may define $B := U\mathrm{diag}(\sqrt{\lambda_0}, \ldots, \sqrt{\lambda_{n-1}}) \in \mathbb{C}^{n \times n}$. Note then that

$$A = U\mathrm{diag}(\sqrt{\lambda_0}, \ldots, \sqrt{\lambda_{n-1}})\mathrm{diag}(\sqrt{\lambda_0}, \ldots, \sqrt{\lambda_{n-1}})U^* = BB^*.$$

The matrix B will be full rank (and therefore invertible) by Exercise 2.7.4 and Lemma 2.9.3.

$\underline{(3) \implies (1)}$: Let $\mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$. Since $B \in \mathbb{C}^{n \times n}$ is invertible/full rank, $B^*\mathbf{x} \neq \mathbf{0}$. Hence,

$$\mathbf{x}^* A \mathbf{x} \;=\; \mathbf{x}^* B B^* \mathbf{x} \;=\; (B^*\mathbf{x})^*(B^*\mathbf{x}) \;=\; \|B^*\mathbf{x}\|_2^2 > 0.$$

□

Similar characterizations exist of PSD matrices, negative definite matrices, and negative semidefinite matrices. We leave them as exercises.

Exercise 2.9.6. *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian. Show that the following are equivalent:*

1. *A is PSD.*
2. *All the eigenvalues of A are nonnegative real numbers.*
3. *There exists a matrix $B \in \mathbb{C}^{n \times n}$ with rank$(B) =$ rank(A) such that $A = BB^*$.*

Exercise 2.9.7. *Prove a variant of Theorem 2.9.4 for negative definite matrices.*

Exercise 2.9.8. *Prove a variant of Theorem 2.9.4 for negative semidefinite matrices.*

2.9.1 The Cholesky Decomposition

Let $B^* \in \mathbb{C}^{n \times n}$ be as in the third part of Theorem 2.9.4 and consider its QR decomposition $B^* = QR$, where $Q \in \mathbb{C}^{n \times n}$ is unitary and $R \in \mathbb{C}^{n \times n}$ is upper triangular. Note that the entries on the diagonal of R must be nonzero since B^* is full rank, but they can be complex. To help make them real we can let $D \in \mathbb{C}^{n \times n}$ be the unitary diagonal matrix

$$D = \mathrm{diag}\left(\left\{\mathbb{e}^{-\mathbb{i}\arg(R_{jj})}\right\}_{j \in [n]}\right)$$

and then write

$$B^* \;=\; QR \;=\; (QD^*)(DR) \;=\; \tilde{Q}\tilde{R},$$

where $\tilde{Q} := QD^*$ is unitary by Exercise 2.6.12, and $\tilde{R} := DR$ is upper triangular with positive real entries on its diagonal ($\tilde{R}_{j,j} = |R_{j,j}| > 0$ for all $j \in [n]$).

Returning to the third part of Theorem 2.9.4 we can see that any positive definite matrix A will have

$$A \;=\; BB^* \;=\; (B^*)^* \, B^* \;=\; (\tilde{Q}\tilde{R})^* \tilde{Q}\tilde{R} \;=\; \tilde{R}^* \tilde{Q}^* \tilde{Q}\tilde{R} \;=\; \tilde{R}^* \tilde{R},$$

since $\tilde{Q}$ is unitary. Note that since $\tilde{R}$ is upper triangular, $\tilde{R}^*$ is lower triangular. We have just proven the following theorem.

Theorem 2.9.5 (Cholesky Decomposition)**.** *Let $A \in \mathbb{C}^{n\times n}$ be positive definite. Then $\exists$ a lower triangular matrix $L \in \mathbb{C}^{n\times n}$ with positive real entries on its diagonal so that $A = LL^*$.*

Exercise 2.9.9. *Let $A \in \mathbb{C}^{n\times n}$ be full rank. Prove that $\exists$ a lower triangular matrix $L \in \mathbb{C}^{n\times n}$ with positive real entries on its diagonal so that $AA^* = LL^*$*

2.10 A Review of the Trace and Determinant Functions

In this section we will rapidly review some important properties of both the trace and the determinant of a matrix. Unlike more introductory texts (see, e.g., [18, 40]), we will give both of these functions a rather cursory treatment focused primarily on their computational aspects. As a result, we encourage any reader with more than a casual interest in either function to consult other sources if the presentation here is found to be lacking.

2.10.1 The Trace of a Matrix

We begin with the more computationally useful of the two functions.

Definition 2.10.1 (Trace)**.** *The trace function,* Trace : $\mathbb{C}^{m\times n} \to \mathbb{C}$, *is defined by*

$$\mathrm{Trace}(A) := \sum_{j \in [\min\{m,n\}]} A_{j,j} = \sum_{j=0}^{\min\{m,n\}-1} A_{j,j}$$

for all $A \in \mathbb{C}^{m\times n}$.

Exercise 2.10.1. *Prove that the trace function is linear. That is, for all $A, B \in \mathbb{C}^{m\times n}$ and $\lambda \in \mathbb{C}$ show that both*

1. $\mathrm{Trace}(A + B) = \mathrm{Trace}(A) + \mathrm{Trace}(B)$, *and*
2. $\mathrm{Trace}(\lambda A) = \lambda \mathrm{Trace}(A)$

hold.

Exercise 2.10.2. *Let* $A \in \mathbb{C}^{m \times n}$*. Prove that* $\mathrm{Trace}(A^*) = \overline{\mathrm{Trace}(A)}$*.*

The following property of trace functions is particularly useful.

Lemma 2.10.2. *Let* $A \in \mathbb{C}^{m \times n}$ *and* $B \in \mathbb{C}^{n \times m}$*. Then* $\mathrm{Trace}(AB) = \mathrm{Trace}(BA)$*.*

Proof. We compute

$$\begin{aligned} \mathrm{Trace}(AB) &= \sum_{j\in[m]} (AB)_{j,j} = \sum_{j\in[m]} \sum_{k\in[n]} A_{j,k} B_{k,j} = \sum_{k\in[n]} \sum_{j\in[m]} B_{k,j} A_{j,k} \\ &= \sum_{k\in[n]} (BA)_{k,k} = \mathrm{Trace}(BA). \end{aligned}$$

□

The trace function has an important relationship to the Frobenius norm of a given matrix $A \in \mathbb{C}^{m \times n}$, allowing one to create an associated inner product. The following lemma explicitly demonstrates the relationship between the Frobenius norm and the trace function.

Lemma 2.10.3. *Let* $A \in \mathbb{C}^{m \times n}$*. Then* $\|A\|_{\mathrm{F}}^2 = \mathrm{Trace}(A^* A) = \mathrm{Trace}(AA^*)$*.*

Proof. We have that

$$\begin{aligned} \|A\|_{\mathrm{F}}^2 &= \sum_{j\in[m]} \sum_{k\in[n]} |A_{j,k}|^2 = \sum_{k\in[n]} \sum_{j\in[m]} A_{j,k} \overline{A_{j,k}} = \sum_{k\in[n]} \sum_{j\in[m]} (A^*)_{k,j} A_{j,k} \\ &= \sum_{k\in[n]} (A^* A)_{k,k} = \mathrm{Trace}(A^* A). \end{aligned}$$

The second equality now follows from Lemma 2.10.2. □

We may now go ahead and define the Frobenius inner product of two matrices $A, B \in \mathbb{C}^{m \times n}$ to be

$$\langle A, B \rangle_{\mathrm{F}} := \mathrm{Trace}(A^* B).$$

To see that this is indeed an inner product, recall that one can always reshape an $m \times n$ matrix into a vector of length mn using, e.g., the vectorization operator $\mathrm{vec} : \mathbb{C}^{m \times n} \to \mathbb{C}^{mn}$ defined for all $\ell \in [mn]$ by $(\mathrm{vec}(A))_\ell = A_{\ell \text{ mod } m, \frac{\ell - \ell \text{ mod } m}{m}}$, where "$\ell$ mod m" is defined for all $\ell \in \mathbb{Z}$ and $m \in \mathbb{N}$ to be the single element contained in the set $\{\ell + km \mid k \in \mathbb{Z}\} \cap [m]$ (or, equivalently, to be the unique value $r \in [m] = \{0, 1, \ldots, m-1\}$ such that $\exists k \in \mathbb{Z}$ satisfying $\ell = r + km$). We then have that

$$\begin{aligned} \langle A, B \rangle_{\mathrm{F}} &= \mathrm{Trace}(A^* B) = \sum_{j\in[n]} \sum_{k\in[m]} (A^*)_{j,k} B_{k,j} = \sum_{j\in[n]} \sum_{k\in[m]} \overline{A_{k,j}} B_{k,j} \\ &= \langle \mathrm{vec}(A), \mathrm{vec}(B) \rangle. \end{aligned}$$

Hence, we can see that the Frobenius inner product is simply the usual Euclidean inner product in disguise.

One benefit of realizing that the Frobenius inner product is indeed an inner product is that we can now apply previously established facts about inner products to learn additional facts about trace functions. For example, we can now apply the Cauchy-Schwarz inequality to see that

$$|\mathrm{Trace}(A^*B)| \;=\; |\langle \mathrm{vec}(A), \mathrm{vec}(B)\rangle| \;\leq\; \|\mathrm{vec}(A)\|_2 \|\mathrm{vec}(B)\|_2 \;=\; \|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}.$$

Lemma 2.10.4 (Cauchy-Schwarz Example for Trace)**.** *Let $A, B \in \mathbb{C}^{m\times n}$. Then*

$$|\mathrm{Trace}(A^*B)| \leq \|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}.$$

We hope that this useful inequality spurs the reader to more deeply appreciate the utility of both abstraction and re-expression. Old facts often return in disguise as new facts, and abstract theory is a channel through which many such elegant re-expressions travel.

2.10.2 The Determinant of a Matrix

The determinant is a function from square matrices into $\mathbb{C}$, denoted herein by $\det : \mathbb{C}^{n\times n} \to \mathbb{C}$. Determinants have many interesting properties and are fundamentally important to both multivariable calculus and differential geometry as well as their relations and descendants. That said, if you ever find yourself actually computing a determinant you're likely doing something wrong. It does, however, happen on occasion....

For a review of the basic properties of determinants we refer the interested reader to, e.g., [18, Chapter 4] and/or [40, Chapter 5]. Herein we will focus on a small subset of their many properties which are of the most value for computing them efficiently.

Theorem 2.10.5 (See, e.g., Appendix C of [20])**.** *Let $A, B \in \mathbb{C}^{n\times n}$. The determinant function $\det : \mathbb{C}^{n\times n} \to \mathbb{C}$ satisfies the following properties:*

1. $\det(A^*) = \overline{\det(A)}$,

2. $\det(AB) = \det(A)\det(B)$, *and*

3. *If A is lower (or upper) triangular, then* $\det(A) = \prod_{j\in[n]} A_{j,j}$.

There are many other useful properties of determinants as well, some of which you may now derive from Theorem 2.10.5.

Exercise 2.10.3. *Use Theorem 2.10.5 to prove the following additional standard facts about determinants.*

1. $\det(I_n) = 1$.

2. *If $A \in \mathbb{C}^{n \times n}$ is invertible, then* $\det\left(A^{-1}\right) = \frac{1}{\det(A)}$.

3. *If $U \in \mathbb{C}^{n \times n}$ is unitary, then* $\det(U) \in \mathbb{C}$ *has magnitude* 1 *so that* $|\det(U)| = 1$.

Exercise 2.10.4. *Prove the following additional standard facts about determinants.*

1. *If $A \in \mathbb{C}^{n \times n}$ is Hermitian, then* $\det(A)$ *is the product of A's eigenvalues.*

2. $\det(\lambda A) = \lambda^n \det(A)$ *for all $A \in \mathbb{C}^{n \times n}$ and $\lambda \in \mathbb{C}$.*

Exercise 2.10.5. *Let $A = QR$ be a QR-decomposition of a full rank matrix $A \in \mathbb{C}^{n \times n}$. Show that* $|A| := |\det(A)| = \prod_{j \in [n]} |R_{j,j}|$.

Exercise 2.10.6. *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian with eigenvalues $\lambda_0, \ldots, \lambda_{n-1} \in \mathbb{R}$. Prove that* $\det(A) = \prod_{j \in [n]} \lambda_j \in \mathbb{R}$.

The second last exercise just above implies a relatively efficient and stable numerical algorithm for computing the absolute determinant of a given matrix.

Chapter 3

Some More Advanced Topics in Linear Algebra

3.1 One Factorization to Rule Them All: The Singular Value Decomposition

The Singular Value Decomposition (SVD) is arguably the most useful fact of Linear Algebra, which is itself arguably the most useful and ubiquitous of mathematical subjects (with respect to computation in particular). The SVD's utility in data analysis is underscored by the fact that it has been (re)discovered at least three times in different scientific communities [44]. In this section we will review the SVD of a given matrix $A \in \mathbb{C}^{m \times n}$. Many sections of the book hereafter will use the SVD repeatedly and often – it is well worth refreshing yourself here, and familiarizing yourself with our notation, before moving on.

Finally, to re-emphasize our statement about linear algebra over the real versus complex numbers from the beginning of Chapter 2, we remind the reader that **replacing the symbol "$\mathbb{C}$" everywhere it appears in this section with an "$\mathbb{R}$" will not affect the correctness of the results herein in any way whatsoever.** In fact, the only cosmetic (and frankly, totally unnecessary) changes that might result by restricting ourselves to $\mathbb{R} \subset \mathbb{C}$ below would be on the order of, e.g., renaming real-valued Hermitian matrices "symmetric matrices", calling the conjugate-transpose of a real-valued matrix just its "transpose", etc..

We will now begin studying the SVD by proving a relatively simple lemma that establishes some notation as well as a large number of potential matrix factorizations which include the SVD as a special case.

Lemma 3.1.1. *Let $A \in \mathbb{C}^{m \times n}$ and $\{\mathbf{w}_0, \cdots, \mathbf{w}_{n-1}\} \subset \mathbb{C}^n$ be an orthonormal basis for $\mathbb{C}^n$.*

Define $s_j := \|A\mathbf{w}_j\|_2$ *(reordering the* $\mathbf{w}_j$*'s as needed so that* $s_0 \geq s_1 \geq \cdots \geq s_{n-1}$*), and let*

$$\mathbf{h}_j := \begin{cases} \mathbf{0} & \text{if } s_j = 0 \\ \frac{1}{s_j} A\mathbf{w}_j \in \mathbb{C}^m & \text{if } s_j \neq 0 \end{cases}. \tag{3.1}$$

Finally, let $W \in \mathbb{C}^{n \times n}$ *be the unitary matrix with* $W_{:,j} = \mathbf{w}_j$ *for all* $j \in [n]$ *and* $H \in \mathbb{C}^{m \times n}$ *be the matrix with* $H_{:,j} = \mathbf{h}_j$ *for all* $j \in [n]$*. Then, we have*

$$A = H \ \mathrm{diag}(s_0, \dots, s_{n-1}) \ W^*$$

where $s_0 \geq s_1 \geq \cdots \geq s_{n-1} \in [0, \infty)$*.*

Proof. We have that

$$\begin{aligned} AW &= A \begin{pmatrix} | & | & & | \\ \mathbf{w}_0 & \mathbf{w}_1 & \cdots & \mathbf{w}_{n-1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ A\mathbf{w}_0 & A\mathbf{w}_1 & \cdots & A\mathbf{w}_{n-1} \\ | & | & & | \end{pmatrix} \\ &= \begin{pmatrix} | & | & & | \\ s_0\mathbf{h}_0 & s_1\mathbf{h}_1 & \cdots & s_{n-1}\mathbf{h}_{n-1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_{n-1} \\ | & | & & | \end{pmatrix} \begin{pmatrix} s_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_{n-1} \end{pmatrix} \\ &= H \ \mathrm{diag}(s_0, \dots, s_{n-1}). \end{aligned}$$

Thus, $A = AWW^* = H \ \mathrm{diag}(s_0, \dots, s_{n-1}) \ W^*$. □

Lemma 3.1.1 already yields a large family of decompositions for any given $A \in \mathbb{C}^{m \times n}$ with several of the structural properties that will ultimately be provided by the singular value decomposition. The next lemma tells us how to choose the orthonormal basis $\{\mathbf{w}_j\}_{j \in [n]}$ of $\mathbb{C}^n$ in order to ensure that the $\mathbf{h}_j$ vectors defined in (3.1) can be used to form a unitary matrix. As a happy coincidence, our choice of $\{\mathbf{w}_j\}_{j \in [n]} \subset \mathbb{C}^n$ will also contain an orthonormal basis for the null space of A as subset of its columns, and guarantee the uniqueness of the ordered s_j values from Lemma 3.1.1.

As we shall see, choosing $\{\mathbf{w}_j\}_{j \in [n]} \subset \mathbb{C}^n$ in Lemma 3.1.1 to be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n \times n}$ is the "correct" choice (at least, if our goal is to try to orthogonalize H as much as possible). And, it's important to note, this choice is always possible by Fact 2.8.6 since A^*A will always be Hermitian no matter what $A \in \mathbb{C}^{m \times n}$ itself looks like. Toward seeing how nicely this works out, let's quickly recall some facts about the four fundamental subspaces of both A and A^*A from Section 2.7. First, if $\mathbf{w}_j \in \mathbb{C}^n$ is an eigenvector of A^*A then $A\mathbf{w}_j = \mathbf{0}$ can only hold if $\mathbf{w}_j \in \mathcal{N}(A) = \mathcal{C}(A^*)^\perp = \mathcal{C}(A^*A)^\perp = \mathcal{N}(A^*A)$ (see, e.g., Lemmas 2.7.2 and 2.7.3). Second, A is rank r if and only if A^*A is rank r (see Theorem 2.7.1 and Lemma 2.7.3). Thus, if A is rank r there will be exactly r orthonormal eigenvectors of A^*A associated with nonzero eigenvalues, and they will span $\mathcal{C}(A^*A) = \mathcal{C}(A^*)$.

Exercise 3.1.1. *Let $A \in \mathbb{C}^{m\times n}$ be rank r and $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n\times n}$. Suppose, w.l.g., that the r orthonormal eigenvectors of A^*A associated with nonzero eigenvalues are $\{\mathbf{w}_j\}_{j\in[r]}$. Show that they are an orthonormal basis of $\mathcal{C}(A^*)$.*

Exercise 3.1.2. *Let $A \in \mathbb{C}^{m\times n}$ be rank r and $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n\times n}$. Prove that $A\mathbf{w}_j = \mathbf{0}$ will hold if and only if $\mathbf{w}_j$ has eigenvalue 0 as an eigenvector of A^*A. Conclude that $A\mathbf{w}_j = \mathbf{0}$ will hold for exactly $n-r$ of the orthonormal eigenvectors of A^*A in $\{\mathbf{w}_j\}_{j\in[n]}$ as a result.*

Next, suppose, w.l.g., that the $n-r$ orthonormal eigenvectors of A^*A above are $\{\mathbf{w}_j\}_{j=r}^{n-1}$. Argue that they are an orthonormal basis of $\mathcal{N}(A)$.

Let $A \in \mathbb{C}^{m\times n}$ be rank r. The next lemma shows that choosing $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ in Lemma 3.1.1 to be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n\times n}$ will result in exactly r nonzero and orthonormal $\mathbf{h}_j$ vectors in (3.1).

Lemma 3.1.2. *Let $A \in \mathbb{C}^{m\times n}$ be rank r. Choose $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ in Lemma 3.1.1 to be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n\times n}$. Then the $\mathbf{h}_j$ vectors defined in* (3.1) *will be such that $\{\mathbf{h}_j\}_{j\in[r]} \subset \mathbb{C}^m$ form an orthonormal basis of $\mathcal{C}(A)$, and $\mathbf{h}_j = \mathbf{0}$ for all $j = r, \dots, n-1$.*

Proof. Exactly r of the $\mathbf{h}_j$ vectors defined in (3.1) will be nonzero by Exercise 3.1.2. Furthermore, these nonzero $\mathbf{h}_j$ vectors will be $\{\mathbf{h}_j\}_{j\in[r]}$ due to the ordering imposed on the $s_j = \|A\mathbf{w}_j\|_2$ values. Finally, each $\mathbf{h}_j \in \mathcal{C}(A)$ will have $\|\mathbf{h}_j\|_2 = 1$ for all $j \in [r]$ by the definition of the $\mathbf{h}_j$ vectors in (3.1). Thus, to finish the proof it suffices by Exercise 2.4.5 to prove that $\{\mathbf{h}_j\}_{j\in[r]}$ is orthogonal.

Let λ_ℓ be the eigenvalue of A^*A associated with an eigenvector $\mathbf{w}_\ell$ for all $0 \le \ell < r$. Considering the inner product of any two nonzero $\mathbf{h}_j$ vectors from (3.1) we have that

$$\langle \mathbf{h}_j, \mathbf{h}_\ell \rangle = \frac{1}{s_j s_\ell}\langle A\mathbf{w}_j, A\mathbf{w}_\ell \rangle = \frac{1}{s_j s_\ell}(A\mathbf{w}_j)^* A\mathbf{w}_\ell = \frac{1}{s_j s_\ell}\mathbf{w}_j^* \left(A^*A\mathbf{w}_\ell\right) = \frac{\lambda_\ell}{s_j s_\ell}\mathbf{w}_j^*\mathbf{w}_\ell = 0$$

whenever $j \neq \ell$ due to the orthonormality of $\{\mathbf{w}_j\}_{j\in[n]}$. Hence, $\{\mathbf{h}_j\}_{j\in[r]}$ is an orthonormal basis of $\mathcal{C}(A)$. □

Exercise 3.1.3. *Let $A \in \mathbb{C}^{m\times n}$ be rank r. Suppose that some choice of the orthonormal basis $\{\mathbf{w}_j\}_{j\in[n]}$ of $\mathbb{C}^n$ in Lemma 3.1.1 results in exactly r nonzero $\mathbf{h}_j$ vectors in* (3.1), *$\{\mathbf{h}_j\}_{j\in[r]}$. Furthermore, suppose that $\{\mathbf{h}_j\}_{j\in[r]}$ is orthonormal. Prove that every $\mathbf{w}_j$ must then be an eigenvector of $A^*A \in \mathbb{C}^{n\times n}$.*

Lemma 3.1.2 combined with Exercise 3.1.3 imply that there is essentially only one way to apply Lemma 3.1.1 so that its H matrix ends up having exactly $r = \text{rank}(A)$ nonzero and orthonormal columns $\{\mathbf{h}_j\}_{j\in[r]}$. We simply must choose $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ to be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of $A^*A \in \mathbb{C}^{n\times n}$. Making that choice, we then have that $\{\mathbf{h}_j\}_{j\in[r]}$ will be an orthonormal basis of $\mathcal{C}(A) \subset \mathbb{C}^m$. We can, therefore, complete $\{\mathbf{h}_j\}_{j\in[r]}$ to be larger orthonormal basis $B = \{\mathbf{h}_j\}_{j\in[r]} \cup \{\mathbf{u}_\ell\}_{\ell=r}^{m-1}$ of all of $\mathbb{C}^m$, where $\{\mathbf{u}_\ell\}_{\ell=r}^{m-1}$ will then be an orthonormal basis of $\mathcal{C}(A)^\perp = \mathcal{N}(A^*)$ by construction.

Let $U \in \mathbb{C}^{m\times m}$ be the unitary matrix with its columns given by

$$U_{:,j} = \begin{cases} \mathbf{h}_j & \text{if } j \in [r] \\ \mathbf{u}_j & \text{otherwise} \end{cases}.$$

In addition, let $V \in \mathbb{C}^{n\times n}$ be the unitary matrix whose columns are our well-chosen $\{\mathbf{w}_j\}_{j\in[n]}$ basis so that $V_{:,j} = \mathbf{w}_j$ for all $j \in [n]$. For our $A \in \mathbb{C}^{m\times n}$ we will then have that

$$\begin{aligned} AV &= \begin{pmatrix} | & | & & | \\ A\mathbf{w}_0 & A\mathbf{w}_1 & \cdots & A\mathbf{w}_{n-1} \\ | & | & & | \end{pmatrix} \\ &= \begin{pmatrix} | & | & & | & | & | & | \\ s_0\mathbf{h}_0 & s_1\mathbf{h}_1 & \cdots & s_{r-1}\mathbf{h}_{r-1} & \mathbf{0} & \cdots & \mathbf{0} \\ | & | & & | & | & | & | \end{pmatrix} \in \mathbb{C}^{m\times n} \qquad (3.2) \\ &= \underbrace{\begin{pmatrix} | & | & & | & | & | & | \\ \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_{r-1} & \mathbf{u}_r & \cdots & \mathbf{u}_{m-1} \\ | & | & & | & | & | & | \end{pmatrix}}_{\in\mathbb{C}^{m\times m}} \underbrace{\begin{pmatrix} \multicolumn{5}{c}{\operatorname{diag}(s_0,\ldots,s_{r-1})} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}}_{\in\mathbb{C}^{m\times n}} \\ &= U\Sigma, \end{aligned}$$

where $\Sigma \in [0,\infty)^{m\times n}$ is a real-valued diagonal matrix whose entries are given by

$$\Sigma_{i,j} = \begin{cases} s_j & \text{if } i = j < r \\ 0 & \text{otherwise} \end{cases}.$$

Multiplying (3.2) through on the right by V^* we finally see that

$$A \;=\; AVV^* \;=\; U\Sigma V^*.$$

Example 3.1.3. *To help the reader digest the abstract computation in* (3.2) *we will perform a specific example of it here. Let* $A = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 2 & 2 \end{pmatrix}$ *so that* $A^*A = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 5 & 3 \\ 1 & 3 & 5 \end{pmatrix}$*. One can then check that*

$$\left\{ \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{-1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix}, \begin{pmatrix} \frac{-2}{\sqrt{6}} \\ \frac{-1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{pmatrix} \right\} \subset \mathbb{R}^3$$

is an orthonormal set of eigenvectors of A^*A (do check this!). Applying A to each of these vectors we obtain

$$A\begin{pmatrix}0\\ \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} = 2\sqrt{2}\begin{pmatrix}0\\1\end{pmatrix},\ A\begin{pmatrix}\frac{1}{\sqrt{3}}\\ \frac{-1}{\sqrt{3}}\\ \frac{1}{\sqrt{3}}\end{pmatrix} = \sqrt{3}\begin{pmatrix}1\\0\end{pmatrix},\ \text{and } A\begin{pmatrix}\frac{-2}{\sqrt{6}}\\ \frac{-1}{\sqrt{6}}\\ \frac{1}{\sqrt{6}}\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix}.$$

Thus, in the terminology of Lemma 3.1.1, we have $s_0 = 2\sqrt{2}$, $s_1 = \sqrt{3}$, $s_2 = 0$, and

$$\mathbf{h}_0 = \begin{pmatrix}0\\1\end{pmatrix},\ \mathbf{h}_1 = \begin{pmatrix}1\\0\end{pmatrix},\ \mathbf{h}_2 = \begin{pmatrix}0\\0\end{pmatrix}.$$

Forming the unitary matrices $U \in \mathbb{R}^{2\times 2}$ and $V \in \mathbb{R}^{3\times 3}$ used in (3.2) *in this case and carrying out the computation to its conclusion we learn that*

$$A = \begin{pmatrix}1 & -1 & 1\\ 0 & 2 & 2\end{pmatrix} = \underbrace{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}_{U}\underbrace{\begin{pmatrix}2\sqrt{2} & 0 & 0\\ 0 & \sqrt{3} & 0\end{pmatrix}}_{\Sigma}\underbrace{\begin{pmatrix}0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & \frac{1}{\sqrt{3}}\\ \frac{-2}{\sqrt{6}} & \frac{-1}{\sqrt{6}} & \frac{1}{\sqrt{6}}\end{pmatrix}}_{V^*}.$$

Exercise 3.1.4. *Repeat the calculation in Example 3.1.3 for the matrix* $A = \begin{pmatrix}1 & 1\\ 1 & 1\\ -1 & 1\end{pmatrix}$.

Formalizing the discussion above allows us to prove the following theorem establishing the existence of the SVD for any matrix $A \in \mathbb{C}^{m\times n}$.

Theorem 3.1.4 (The Full Singular Value Decomposition)**.** *Every rank r matrix $A \in \mathbb{C}^{m\times n}$ can be decomposed into $A = U\Sigma V^*$ where*

1. *$U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are both unitary, and*
2. *$\Sigma \in [0,\infty)^{m\times n}$ is a unique diagonal matrix with entries*

$$\Sigma_{i,j} = \begin{cases}\sigma_j(A) & \text{if } i = j\\ 0 & \text{otherwise}\end{cases}$$

satisfying $\sigma_0(A) \geq \sigma_1(A) \geq \cdots \geq \sigma_{r-1}(A) > 0 = \sigma_r(A) = \cdots = \sigma_{\min\{m,n\}-1}(A)$.

Here the j^{th}-largest diagonal entry of the diagonal matrix Σ, $\sigma_j(A) \in [0,\infty)$, is called the j^{th} **singular value of** *A. Similarly, given a valid SVD of A, $A = U\Sigma V^*$, the vectors $\mathbf{u}_j = U_{:,j} \in \mathbb{C}^m$ and $\mathbf{v}_j = V_{:,j} \in \mathbb{C}^n$ are called the j^{th}* **left and right (respectively) singular vectors of (the SVD of)** *A.*[1]

[1] These slightly awkward names for $\mathbf{u}_j = U_{:,j} \in \mathbb{C}^m$ and $\mathbf{v}_j = V_{:,j} \in \mathbb{C}^n$ are due to the fact that these vectors are not generally unique for a given matrix A. Note that there will be many unitary U and V matrix pairs that work as part of a valid SVD of A, especially when there are repeated singular values.

Exercise 3.1.5. *Let $A \in \mathbb{C}^{m \times n}$ have the full SVD $A = U\Sigma V^*$. Set $r = \operatorname{rank}(A)$. Show that*

$$A = \sum_{j \in [r]} \sigma_j(A) \mathbf{u}_j \mathbf{v}_j^* \tag{3.3}$$

where $\sigma_j(A)$ is the j^{th} singular value of A, and $\mathbf{u}_j = U_{:,j} \in \mathbb{C}^m$, $\mathbf{v}_j = V_{:,j} \in \mathbb{C}^n$ are the , j^{th} left/right singular vectors of the SVD of A. (Hint: Consider using Exercise 2.6.9.)

One can now prove the following corollary from Theorem 3.1.4 via an argument analogous to the one used to derive Corollary 2.8.9 from Theorem 2.8.7 (or, alternatively, by using (3.3) from Exercise 3.1.5 to build the new factorization more directly).

Corollary 3.1.5 (The Compact Singular Value Decomposition)**.** *Every rank r matrix $A \in \mathbb{C}^{m \times n}$ can be decomposed into $A = U\Sigma V^*$ where*

1. *$U \in \mathbb{C}^{m \times r}$ and $V \in \mathbb{C}^{n \times r}$ are both orthonormal matrices, and*
2. *$\Sigma = \operatorname{diag}(\sigma_0(A), \dots, \sigma_{r-1}(A)) \in [0, \infty)^{r \times r}$ is a unique diagonal matrix containing the r nonzero singular values of A ordered so that $\sigma_0(A) \geq \sigma_1(A) \geq \cdots \geq \sigma_{r-1}(A) > 0$.*

Exercise 3.1.6. *Prove Corollary 3.1.5.*

However one proves Theorem 3.1.4 and Corollary 3.1.5, the *uniqueness* of the singular values of a matrix $A \in \mathbb{C}^{m \times n}$ ultimately follows from the fact that they must always be the square roots of the eigenvalues of $A^*A \in \mathbb{C}^{n \times n}$ (and $AA^* \in \mathbb{C}^{m \times m}$). For this reason (in addition to several others), we will now briefly review the properties that any valid SVD of a matrix A must share with the spectral decompositions of both A^*A and AA^*.

3.1.1 The Relationship to the Spectral Decompositions of A^*A and AA^*

Let $A = U\Sigma V^*$ be a valid full SVD of a rank r matrix $A \in \mathbb{C}^{m \times n}$ (i.e., so that $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are both unitary, and $\Sigma \in [0, \infty)^{m \times n}$ is a diagonal matrix satisfying $\Sigma_{0,0} \geq \cdots \geq \Sigma_{r-1,r-1} > \Sigma_{r,r} = \cdots = \Sigma_{q-1,q-1} = 0$, where $q = \min\{m, n\}$). Notice that then

$$A^*A \;=\; (U\Sigma V^*)^*(U\Sigma V^*) \;=\; V\Sigma^* U^* U \Sigma V^* \;=\; V(\Sigma^*\Sigma)V^*,$$

where $D = \Sigma^*\Sigma \in [0, \infty)^{n \times n}$ is a diagonal matrix with $D_{0,0} = \Sigma_{0,0}^2 \geq \cdots \geq D_{r-1,r-1} = \Sigma_{r-1,r-1}^2 > D_{r,r} = \cdots = D_{n-1,n-1} = 0$. As a consequence, we can see that every column $\mathbf{v}_j = V_{:,j}$ of V will be an eigenvector of A^*A with eigenvalue $D_{j,j}$ since

$$A^*A\mathbf{v}_j \;=\; V(\Sigma^*\Sigma)V^*\mathbf{v}_j \;=\; V(\Sigma^*\Sigma)\mathbf{e}_j \;=\; VD_{j,j}\mathbf{e}_j \;=\; D_{j,j}\mathbf{v}_j.$$

Thus, $D_{j,j}$ must be the j^{th} largest eigenvalue of $A^*A \in \mathbb{C}^{n \times n}$. Given that the eigenvalues of A^*A are both unique (with potential repetitions since they are the zeros of the characteristic polynomial of A^*A – see, e.g., [20, Chapter 10]), and always nonnegative real numbers (see Exercise 2.8.6), this further implies that each $\Sigma_{j,j} = \sqrt{D_{j,j}}$ is also uniquely determined by A. Hence, we'll call the value that $\Sigma_{j,j}$ must always take in any valid full SVD of A "$\sigma_j(A)$", and will later discuss it even in the absence of a particular SVD of A.

Exercise 3.1.7. *Let $A = U\Sigma V^*$ be a valid full SVD of a rank r matrix $A \in \mathbb{C}^{m\times n}$. Show that $\Sigma_{j,j}$ must always equal the square-root of the $j^{\rm th}$ largest eigenvalue of $AA^* \in \mathbb{C}^{m\times m}$. Conclude that the nonzero eigenvalues of $AA^* \in \mathbb{C}^{m\times m}$ must always match the nonzero eigenvalues of $A^*A \in \mathbb{C}^{n\times n}$.*

The following result can be proven by carefully considering the discussion so far.

Theorem 3.1.6. *Let $A = U\Sigma V^*$ be a valid full SVD of a rank r matrix $A \in \mathbb{C}^{m\times n}$. The following statements must hold:*

1. *The r nonzero singular values of A are exactly the square roots of the positive eigenvalues of $A^*A \in \mathbb{C}^{n\times n}$ and $AA^* \in \mathbb{C}^{m\times m}$.*

2. *The first r columns of $U \in \mathbb{C}^{m\times m}$ are an orthonormal basis for the column space of A, $\mathcal{C}(A) \subset \mathbb{C}^m$.*

3. *The last $m-r$ columns of $U \in \mathbb{C}^{m\times m}$ form an orthonormal basis for the null space of A^*, $\mathcal{N}(A^*) \subset \mathbb{C}^m$.*

4. *The first r columns of $V \in \mathbb{C}^{n\times n}$ form an orthonormal basis for the column space of A^*, $\mathcal{C}(A^*) \subset \mathbb{C}^n$.*

5. *The last $n-r$ columns of $V \in \mathbb{C}^{n\times n}$ form an orthonormal basis for the null space of A, $\mathcal{N}(A) \subset \mathbb{C}^n$.*

6. *If $m = n$ and A is Hermitian, then A will have λ as an eigenvalue if and only if there exists a $j \in [n]$ such that*

   - *$|\lambda|$ is the $j^{\rm th}$ singular value of A (i.e., $\sigma_j = |\lambda|$),*
   - *the $j^{\rm th}$ column of V, $\mathbf{v}_j \in \mathbb{C}^n$, is an eigenvector of A associated with λ, and*
   - *the $j^{\rm th}$ column of $U = \operatorname{sign}(\lambda)\mathbf{v}_j$.*

Exercise 3.1.8. *Prove Theorem 3.1.6.*

Exercise 3.1.9. *Let $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ both be unitary, $A \in \mathbb{C}^{m\times n}$, and $q := \min\{m,n\}$. Show that $\sigma_j(UA) = \sigma_j(A) = \sigma_j(AV)$ holds for all $j \in [q]$.*

Exercise 3.1.10. *Let $\alpha, \beta \in \mathbb{Z} \setminus \{0\}$. The $\frac{\alpha}{\beta}$-power of a full rank matrix $A \in \mathbb{C}^{n\times n}$ is a matrix $B \in \mathbb{C}^{n\times n}$ with the property that $B^\beta = A^\alpha$ (e.g., when $\beta = 2$ and $\alpha = 1$ then B is called the matrix square root of A). Prove that there always exists a unitary matrix $W \in \mathbb{C}^{n\times n}$ such that any desired $\frac{\alpha}{\beta}$-power of AW exists. When can W simply be the identity? How can one compute such a B and W for any given $A \in \mathbb{C}^{n\times n}$?*

As Theorem 3.1.6 hopefully makes clear, a SVD of A conveniently encodes just about any standard information you might want to know about A. It is a commonly computed decomposition as a result. Numerically, a SVD of a small to moderately sized matrix $A \in \mathbb{C}^{m\times n}$ can be efficiently computed using a variety of standard methods (depending on how, e.g., m compares in size to n). We refer the interested reader to numerical linear algebra texts such as [47, Lecture 31] or [16, Chapter 5] for details. For an extremely large matrix $A \in \mathbb{C}^{m\times n}$ that might not be (able to be) stored on a single machine, however, one might have to utilize a distributed/incremental SVD algorithm instead (see, e.g., [8, 9, 27]).

3.2 The SVD and the Moore–Penrose Inverse of a Matrix

Note that every matrix $A \in \mathbb{C}^{m\times n}$ is a linear bijection from $\mathcal{C}(A^*)$ onto $C(A)$. Hence, $A : \mathcal{C}(A^*) \to C(A)$ always has a linear inverse with the same rank as A called the **Moore–Penrose (or, pseudo)inverse of** A, denoted by $A^\dagger : C(A) \to \mathcal{C}(A^*)$. Furthermore, a compact SVD of $A^\dagger \in \mathbb{C}^{n\times m}$ can always be obtained from a compact SVD of A.

Let $A = U\Sigma V^*$ be a compact SVD of a rank r matrix $A \in \mathbb{C}^{m\times n}$ so that $U \in \mathbb{C}^{m\times r}$ and $V \in \mathbb{C}^{n\times r}$ are orthonormal matrices, and $\Sigma = \operatorname{diag}(\sigma_0(A), \dots, \sigma_{r-1}(A)) \in [0,\infty)^{r\times r}$ is invertible (due to $\sigma_0(A) \geq \cdots \geq \sigma_{r-1}(A) > 0$). One can now see that

$$A^\dagger = V\Sigma^{-1}U^* \tag{3.4}$$

must hold. To understand why, recall that the orthogonal projections $P_{\mathcal{C}(A)}$ and $P_{\mathcal{C}(A^*)}$ act as the identities on $\mathcal{C}(A)$ and $\mathcal{C}(A^*)$, respectively (see Theorem 2.6.10). And, e.g.,

$$A^\dagger A \;=\; (V\Sigma^{-1}U^*)(U\Sigma V^*) \;=\; V\Sigma^{-1}I_r\Sigma V^* \;=\; VV^* = P_{\mathcal{C}(A^*)}$$

by (2.14) and part (4) of Theorem 3.1.6. Hence, $A^\dagger : C(A) \to \mathcal{C}(A^*)$ from (3.4) is indeed the left inverse of $A : \mathcal{C}(A^*) \to C(A)$. A similar calculation shows that $AA^\dagger = P_{\mathcal{C}(A)}$ also holds.

Exercise 3.2.1. *Let $A = U\Sigma V^*$ be a compact SVD of a rank r matrix $A \in \mathbb{C}^{m\times n}$. Show that $A^\dagger$ from* (3.4) *satisfies $AA^\dagger = P_{\mathcal{C}(A)}$.*

Exercise 3.2.2. *Suppose that $A \in \mathbb{C}^{n\times n}$ is full rank (so that* $\operatorname{rank}(A) = n$*). Show that $A^\dagger = A^{-1} \in \mathbb{C}^{n\times n}$ in this case.*

The exercise directly above demonstrates that $A^\dagger$ is a *strict generalization* of the "usual" matrix inverse A^{-1}. As a result, in some sense we always should (and really always effectively do) work with $A^{-1} := A^\dagger$ when thinking about inverting a matrix of any size.

Exercise 3.2.3. *Suppose that $A \in \mathbb{C}^{n \times n}$ is full rank (so that $\operatorname{rank}(A) = n$). Show that $\sigma_0\left(A^{-1}\right) = \frac{1}{\sigma_{n-1}(A)}$. More generally, show that $\sigma_j\left(A^{-1}\right) = \frac{1}{\sigma_{n-1-j}(A)}$ for all $j \in [n]$.*

Exercise 3.2.4. *Let $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ have rank n. Show that $A^\dagger = (A^*A)^{-1}A^*$.*

3.3 Some Important Properties of Singular Values

If we have not yet convinced you that the SVD is potentially interesting and useful, we will try again here by showing that two of the most commonly used matrix norms from Section 2.2.3 are closely related to the singular values of a given matrix.

Singular Values and the Frobenius Norm

Given $A \in \mathbb{C}^{m \times n}$ recall that $\|A\|_{\mathrm{F}} = \sqrt{\sum_{\ell,j} |A_{\ell,j}|^2}$. Let $A = U\Sigma V^*$ be a full SVD of A, and set $q := \min\{m, n\}$. Computing the squared Frobenius norm of A via its SVD we can see that

$$\begin{aligned}\|A\|_{\mathrm{F}}^2 &= \|U\Sigma V^*\|_{\mathrm{F}}^2 = \sum_{j\in[n]} \|\left(U\Sigma V^*\right)_{:,j}\|_2^2 = \sum_{j\in[n]} \|U\left(\Sigma V^*\right)_{:,j}\|_2^2 \\ &= \sum_{j\in[n]} \|\left(\Sigma V^*\right)_{:,j}\|_2^2 = \|\Sigma V^*\|_{\mathrm{F}}^2\end{aligned}$$

by Exercise 2.6.13 since U is unitary. Continuing, we can further see that since $\|A\|_{\mathrm{F}} = \|A^*\|_{\mathrm{F}}$ holds for all $A \in \mathbb{C}^{m \times n}$ we also have that

$$\|A\|_{\mathrm{F}}^2 = \|V\Sigma^*\|_{\mathrm{F}}^2 = \sum_{j\in[m]} \|V\left(\Sigma^*\right)_{:,j}\|_2^2 = \sum_{j\in[m]} \|\Sigma^*_{:,j}\|_2^2 = \sum_{j\in[q]} (\sigma_j(A))^2. \tag{3.5}$$

We will see that (3.5) has several important implications in later sections.

Exercise 3.3.1. *Let $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ both be unitary. Show that $\|UA\|_{\mathrm{F}} = \|A\|_{\mathrm{F}} = \|AV\|_{\mathrm{F}}$ holds for all $A \in \mathbb{C}^{m \times n}$.*

Singular Values and the (ℓ^2, ℓ^2)-Operator Norm

Given $A \in \mathbb{C}^{m \times n}$ recall that $\|A\|_{2\to2} = \max\limits_{\mathbf{x}\in\mathbb{C}^n \text{ s.t. } \|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2$. Let $A = U\Sigma V^*$ be a full SVD of A, and set $q := \min\{m, n\}$. Computing the (ℓ^2, ℓ^2)-operator norm of A via its SVD we can see that

$$\|A\|_{2\to2} = \max_{\mathbf{x}\in\mathbb{C}^n \text{ s.t. } \|\mathbf{x}\|_2=1} \|U\Sigma V^*\mathbf{x}\|_2 = \max_{\mathbf{x}\in\mathbb{C}^n \text{ s.t. } \|\mathbf{x}\|_2=1} \|\Sigma V^*\mathbf{x}\|_2$$

by Exercise 2.6.13 since U is unitary. Furthermore, since V is also unitary its columns form an orthonormal basis of $\mathbb{C}^n$ so that every $\mathbf{x} \in \mathbb{C}^n$ with $\|\mathbf{x}\|_2 = 1$ can be written as $\mathbf{x} = \sum_{j=1}^{n} \alpha_j V_{:,j}$ where $\|\boldsymbol{\alpha}\|_2 = \|\mathbf{x}\|_2 = 1$ (see Theorem 2.3.9). Thus, continuing we can see that

$$\begin{aligned} \|A\|_{2\to 2} &= \max_{\boldsymbol{\alpha}\in\mathbb{C}^n \text{ s.t. } \|\boldsymbol{\alpha}\|_2=1} \left\| \Sigma V^* \left(\sum_{j=1}^{n} \alpha_j V_{:,j} \right) \right\|_2 = \max_{\boldsymbol{\alpha}\in\mathbb{C}^n \text{ s.t. } \|\boldsymbol{\alpha}\|_2=1} \left\| \Sigma \left(\sum_{j=1}^{n} \alpha_j \mathbf{e}_j \right) \right\|_2 \\ &= \max_{\boldsymbol{\alpha}\in\mathbb{C}^n \text{ s.t. } \|\boldsymbol{\alpha}\|_2=1} \|\Sigma \boldsymbol{\alpha}\|_2 = \max_{\boldsymbol{\alpha}\in\mathbb{C}^n \text{ s.t. } \|\boldsymbol{\alpha}\|_2=1} \sqrt{\sum_{j\in[q]} |\alpha_j|^2 (\sigma_j(A))^2}. \end{aligned}$$

Recalling that $\sigma_0(A) \geq \sigma_1(A) \geq \cdots \geq \sigma_{q-1}(A)$ we can now see that this last expression is always maximized when $|\alpha_0| = 1$. Hence,

$$\|A\|_{2\to 2} = \sigma_0(A). \tag{3.6}$$

We will see that (3.6) also has several important implications in later sections.

Exercise 3.3.2. *Let $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ both be unitary. Show that $\|UA\|_{2\to 2} = \|A\|_{2\to 2} = \|AV\|_{2\to 2}$ holds for all $A \in \mathbb{C}^{m\times n}$.*

Exercise 3.3.3. *Let $A \in \mathbb{C}^{m\times n}$ and set $q := \min\{m, n\}$. Prove that $\|A\|_{2\to 2} \leq \|A\|_{\mathrm{F}} \leq \sqrt{q}\|A\|_{2\to 2}$ always holds. For what type of matrices will $\|A\|_{2\to 2} = \|A\|_{\mathrm{F}}$ hold? For what type of matrices will $\|A\|_{\mathrm{F}} = \sqrt{q}\|A\|_{2\to 2}$ hold?*

3.3.1 Singular Value Inequalities for Sums and Products of Matrices

Now that we have seen a few reasons why we might want to compute a singular value decomposition of a matrix (e.g., to compute its Moore–Penrose inverse, or its (ℓ^2, ℓ^2)-operator norm), it's worth considering how robust a matrix SVD actually is to small errors. Imagine, for example, that we want to compute the singular values of a matrix $A \in \mathbb{C}^{m\times n}$ on a digital computer. We will encounter potential problems immediately since, unfortunately, we probably can't even store A exactly on our computer! Instead, we will actually store $A + E$, where $E \in \mathbb{C}^{m\times n}$ contains all the round-off errors that result form representing each entry of A with a finite number of binary digits (i.e., bits). Given that we can (at best) then compute the singular values of $A + E$ instead of A, it'd be good to know how close the singular values of $A + E$ are to the true singular values of A we actually want. If, e.g., E has a small Frobenius norm (and, therefore, small singular values by (3.5)) we want to make sure that $\sigma_j(A + E) \approx \sigma_j(A)$ holds for all relevant j. We will now state some very useful singular value inequalities which effectively show that singular values are indeed robust to small perturbations in this way (both additive and multiplicative).

Theorem 3.3.1 (See Theorem 3.3.16 in [24])**.** *Let $A, B \in \mathbb{C}^{m\times n}$ and $q = \min\{m, n\}$. Then* *(a)* $\sigma_{j+k}\left(A+B\right) \leq \sigma_j\left(A\right) + \sigma_k\left(B\right)$, *and*

(b) $\sigma_{j+k}\left(AB^*\right) \leq \sigma_j\left(A\right)\sigma_k\left(B\right)$

for all $j, k \in [q]$ *such that* $j+k \in [q]$*. In particular,*

(c) $\left|\sigma_j\left(A+B\right) - \sigma_j\left(A\right)\right| \leq \sigma_0\left(B\right) \quad \forall j \in [q]$, *and*

(d) $\sigma_j(AB^*) \leq \sigma_j\left(A\right)\sigma_0\left(B\right) \qquad \forall j \in [q]$.

Exercise 3.3.4. *Let* $B \in \mathbb{C}^{m\times n}$ *and* $q = \min\{m,n\}$*. Prove that* $\sigma_j(-B) = \sigma_j(B) = \sigma_j(B^*)$ *holds for all* $j \in [q]$*.*

Exercise 3.3.5. *Use parts (a) and (b) of Theorem 3.3.1 to prove parts (c) and (d).*

Looking at Theorem 3.3.1 (c) one can see that if $B = E$ has a small largest singular value, $\sigma_0\left(E\right)$, then we will indeed have $\sigma_j(A+E) \approx \sigma_j(A)$ for all $j \in [q]$. Furthermore, one can also use similar ideas to see, e.g., that small perturbations to the entries of A won't influence how it behaves as a linear function too much either. This means that matrices can be applied as linear functions on digital computers without distorting their outputs too much.

Example 3.3.2. *Suppose that* $A \in \mathbb{C}^{m\times n}$ *is stored on a digital computer as* $\tilde{A} = A + E$*, where* $E \in \mathbb{C}^{m\times n}$ *is, e.g., a round-off error matrix with* $|E_{i,j}| \leq \epsilon$ *for all* i,j*. How much can* $\tilde{A}\mathbf{x}$ *differ from* $A\mathbf{x}$ *on a worst-case input vector* $\mathbf{x} \in \mathbb{C}^n$*?*

To answer this question we will upper bound $\left\|A\mathbf{x} - \tilde{A}\mathbf{x}\right\|_2$*. Considering this error we can see that*

$$\begin{aligned}\left\|A\mathbf{x} - \tilde{A}\mathbf{x}\right\|_2 &= \|\mathbf{x}\|_2\left\|(A-\tilde{A})\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right\|_2 = \|\mathbf{x}\|_2\left\|E\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right)\right\|_2 \\ &\leq \|\mathbf{x}\|_2\|E\|_{2\to 2} = \|\mathbf{x}\|_2\sigma_0(E).\end{aligned}$$

If we want an upper bound in terms of ϵ *we can now use the fact that* $\|E\|_{2\to 2} \leq \|E\|_{\mathrm{F}}$ *always holds (see Exercise 3.3.3) to get that*

$$\left\|A\mathbf{x} - \tilde{A}\mathbf{x}\right\|_2 \leq \|\mathbf{x}\|_2\|E\|_{\mathrm{F}} \leq \epsilon\|\mathbf{x}\|_2\sqrt{mn}.$$

Thus, the error is will always be small in ℓ^2*-norm as long as* ϵ *is small compared to* $\|\mathbf{x}\|_2\sqrt{mn}$*.*

The following two singular value inequalities are also useful in a variety of applications.

Lemma 3.3.3 (A Slight Generalization of Theorem 3.3.1 Part (d))**.** *Let* $A \in \mathbb{C}^{m\times n}$ *and* $B \in \mathbb{C}^{n\times p}$*. Then,*

$$\sigma_j\left(AB\right) \leq \min\left\{\sigma_j\left(A\right)\sigma_0\left(B\right), \sigma_j\left(B\right)\sigma_0\left(A\right)\right\} \quad \forall j \in [\min\{n,m,p\}].$$

Proof. Suppose, without loss of generality, that $m \leq p$ (else, we may instead apply the argument below to $\sigma_j\left((AB)^*\right) = \sigma_j\left(B^*A^*\right)$ using that $\sigma_j\left(AB\right) = \sigma_j\left((AB)^*\right)$). Since $m \leq p$ we can project all of $\mathbb{C}^m$ into $\mathbb{C}^p$ with $Q = \begin{pmatrix} I_m \\ \cdots \mathbf{0} \cdots \end{pmatrix} \in \mathbb{C}^{p \times m}$. Further, we may note that

$$\sigma_j\left(QA\right) = \sigma_j\left(A\right) \text{ and } \sigma_j\left(QAB\right) = \sigma_j\left(AB\right) \quad \forall j \in [\min\{m, n\}] = [\min\{n, m, p\}].$$

Applying part (d) of Theorem 3.3.1 we can now see that both

$$\sigma_j\left(AB\right) \; = \; \sigma_j\left(QAB\right) \; \leq \; \sigma_j\left(QA\right)\sigma_0\left(B^*\right) \; = \; \sigma_j\left(A\right)\sigma_0\left(B\right)$$

and

$$\sigma_j\left(AB\right) \; = \; \sigma_j\left(QAB\right) \; = \; \sigma_j\left(B^*\left(QA\right)^*\right) \; \leq \; \sigma_j\left(B^*\right)\sigma_0\left(QA\right) \; = \; \sigma_j\left(B\right)\sigma_0\left(A\right)$$

hold. The result follows. □

The next lemma provides lower bounds for the singular values of matrix products in special circumstances.

Lemma 3.3.4. *Let $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{n \times p}$ be such that*

1.) B has a full SVD $B = U\Sigma V^*$, and

2.) AU has rank $r = n$ with a compact SVD $AU = Q\tilde{\Sigma}P^*$.

Then,

$$\sigma_j\left(AB\right) \geq \sigma_r\left(AU\right)\sigma_j\left(B\right) \qquad \forall j \in [r].$$

Proof. Let $j \in [r]$. Noting that P is unitary since $r = n$, we can see that

$$\sigma_j\left(B\right) \; = \; \sigma_j\left(\Sigma V^*\right) \; = \; \sigma_j\left(P\tilde{\Sigma}^{-1}\tilde{\Sigma}P^*\Sigma V^*\right) \; \leq \; \sigma_0\left(P\tilde{\Sigma}^{-1}\right) \cdot \sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right)$$

by Lemma 3.3.3. Furthermore, $\sigma_0\left(P\tilde{\Sigma}^{-1}\right) = \sigma_0\left(\tilde{\Sigma}^{-1}\right) = \frac{1}{\sigma_r\left(\tilde{\Sigma}\right)} = \frac{1}{\sigma_r(AU)}$. Hence,

$$\sigma_j\left(B\right) \; \leq \; \frac{\sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right)}{\sigma_r\left(AU\right)} \; \Longrightarrow \; \sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right) \geq \sigma_r\left(AU\right)\sigma_j\left(B\right). \tag{3.7}$$

Finally, since $m \geq r = n$ we can see that $Q^*Q = I_n$ so that

$$\begin{aligned} \sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right) &= \sigma_j\left(Q^*Q\tilde{\Sigma}P^*\Sigma V^*\right) \; = \; \sigma_j\left(Q^*AU\Sigma V^*\right) \; = \; \sigma_j\left(Q^*AB\right) \\ &\leq \sigma_j\left(AB\right)\sigma_0\left(Q^*\right) \; = \; \sigma_j\left(AB\right) \end{aligned} \tag{3.8}$$

by Lemma 3.3.3. Combing (3.7) and (3.8) now finishes the proof. □

Though perturbation bounds for singular values such as those in Theorem 3.3.1 are both more commonly used and far more robust, it's also worth knowing about the existence of similarly useful perturbation theory for singular vectors/subspaces as well. We urge the interested reader to peruse, e.g., [42, 43] to get a good overview of these results.

3.4 The Optimal Rank-s Approximation A_s of a Matrix A

Recalling Section 1.2.3, suppose that we have trained a deep FNN resulting in a large number of huge weight matrices, $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$, where both d_j and d_{j-1} are "big" for most $j \in [L]$. Our goal is to compress these huge weight matrices as much as possible so that our FNN is easier to store. Simultaneously, we want to accurately preserve each weight matrix as a linear function so that our overall FNN still does what we need it to do after compression. Motivated by, e.g., Section 2.5 we can aim to accomplish our goal by approximating each huge weight matrix W_j by a new low-rank matrix $\tilde{W}_j$ that we can then store in an optimally compressed form. At the same time, Example 3.3.2 implies that it would also be helpful to, e.g., produce $\tilde{W}_j$ in a way that reduces the value of $\left\|W_j - \tilde{W}_j\right\|_{2\to 2} = \sigma_0\left(W_j - \tilde{W}_j\right)$ as much as possible since doing so will help to keep $W_j\mathbf{x} \approx \tilde{W}_j\mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^{d_{j-1}}$.

These considerations collectively suggest the following two step low-rank compression approach for our FNN weight matrices:

1. Approximate each of $W_0, \dots, W_L$ using low-rank matrices $\tilde{W}_0, \dots, \tilde{W}_L$ so that, e.g., $\left\|W_j - \tilde{W}_j\right\|_{2\to 2}$ is small for all $j \in [L]$, and then
2. store $\tilde{W}_0, \dots, \tilde{W}_L$ in a compressed format.

We have already discussed step 2 above in Section 2.5, so we will focus on step 1 here. As we shall see, the SVD is once again extremely useful in this setting, and ultimately allows us to accomplish step 1 in an optimal way.

Let $A \in \mathbb{C}^{m\times n}$ be an arbitrary (e.g., full rank) matrix, and suppose that we want to approximate A with a rank s matrix $A_s \in \mathbb{C}^{m\times n}$ that, e.g., minimizes $\|A - A_s\|_{2\to 2}$ over all possible choices of rank s matrices in $\mathbb{C}^{m\times n}$ so that

$$\|A - A_s\|_{2\to 2} = \inf_{\text{rank } s\ B\in\mathbb{C}^{m\times n}} \|A - B\|_{2\to 2}.$$

To find $A_s \in \mathbb{C}^{m\times n}$, let $A = U\Sigma V^*$ be a full SVD of A and recall that we can then always write

$$A = \sum_{j\in[q]} \sigma_j(A)\mathbf{u}_j\mathbf{v}_j^*,$$

where $q = \min\{m,n\}$, $\mathbf{u}_j = U_{:,j}$, and $\mathbf{v}_j = V_{:,j}$ (see Exercise 3.1.5). We claim that

$$A_s := \sum_{j\in[s]} \sigma_j(A)\mathbf{u}_j\mathbf{v}_j^* \tag{3.9}$$

is then an optimal rank s approximation to A with respect to both the Frobenius and the (ℓ^2, ℓ^2)-operator norms.

Exercise 3.4.1. *Let $A \in \mathbb{C}^{m\times n}$, $q = \min\{m,n\}$, and $A_s \in \mathbb{C}^{m\times n}$ be as in* (3.9)*. Show that $\sigma_j\left(A - A_s\right) = \sigma_{j+s}\left(A\right)$ for all $j \in [q-s]$, and that $\sigma_j\left(A - A_s\right) = 0$ for all $q - s \le j < q$.*

3.4.1 Optimality of A_s in the Frobenius and (ℓ^2, ℓ^2)-Operator Norms

Observe that for the Frobenius norm we have

$$\|A - A_s\|_{\mathrm{F}}^2 \;=\; \left\| \sum_{j=s}^{q} \sigma_j(A) \mathbf{u}_j \mathbf{v}_j^* \right\|_{\mathrm{F}}^2 \;=\; \sum_{j=s}^{q-1} \sigma_j^2(A) \tag{3.10}$$

by (3.5) and Exercise 3.4.1. The next theorem shows that this approximation error is minimal.

Theorem 3.4.1. *Let $A, B \in \mathbb{C}^{m \times n}$, $q = \min\{m, n\}$, and $A_s \in \mathbb{C}^{m \times n}$ be as in* (3.9)*. Furthermore, suppose that be B is rank s. Then*

$$\|A - B\|_{\mathrm{F}} \geq \|A - A_s\|_{\mathrm{F}}.$$

That is, A_s is a best rank s approximation to A with respect to Frobenius norm error.

Proof. Note that $\sigma_s(B) = 0$ by Theorem 3.1.4 since B is rank s. Thus, Theorem 3.3.1 implies that

$$\sigma_{j+s}(A) \;=\; \sigma_{j+s}\left((A - B) + B\right) \;\leq\; \sigma_j(A - B) + \sigma_s(B) \;=\; \sigma_j(A - B)$$

for all $j \in [q - s]$. As a result, (3.5) and (3.10) now reveal that

$$\begin{aligned} \|A - B\|_{\mathrm{F}}^2 \;&=\; \sum_{j \in [q]} \sigma_j^2(A - B) \;=\; \sum_{j \in [q-s]} \sigma_j^2(A - B) + \sum_{j \geq q-s} \sigma_j^2(A - B) \\ &\geq\; \sum_{j \in [q-s]} \sigma_{j+s}^2(A) \;=\; \|A - A_s\|_{\mathrm{F}}^2. \end{aligned}$$

Hence, A_s achieves the smallest possible Frobenius norm approximation error achievable by any rank s matrix. □

Now observe that for the (ℓ^2, ℓ^2)-operator norm we have

$$\|A - A_s\|_{2\to 2} \;=\; \left\| \sum_{j=s}^{q} \sigma_j(A) \mathbf{u}_j \mathbf{v}_j^* \right\|_{2\to 2} \;=\; \sigma_s(A) \tag{3.11}$$

by (3.6) and Exercise 3.4.1. The next theorem shows that this approximation error is also minimal.

Theorem 3.4.2. *Let $A, B \in \mathbb{C}^{m \times n}$, $q = \min\{m, n\}$, and $A_s \in \mathbb{C}^{m \times n}$ be as in* (3.9)*. Furthermore, suppose that be B is rank s. Then*

$$\|A - B\|_{2\to 2} \geq \|A - A_s\|_{2\to 2}.$$

That is, A_s is a best rank s approximation to A with respect to (ℓ^2, ℓ^2)-operator norm error.

Proof. Since B is rank s we can write it in terms of a QR decomposition $B = QR$, where $Q \in \mathbb{C}^{m \times s}$ and $R \in \mathbb{C}^{s \times n}$. Similarly, let $A = U\Sigma V^*$ be a full SVD of A. Since V is unitary, $\mathscr{L} = \operatorname{span}\{V_{:,0}, \dots, V_{:,s}\} \subset \mathbb{C}^n$ has dimension $s+1$. Also, we know that $\mathcal{C}(R^*)^\perp = \mathcal{N}(R)$ has dimension $n - s$ by (the discussion around) Lemma 2.7.2. Hence, it must be the case that

$$\operatorname{span}\{V_{:,0}, \dots, V_{:,s}\} \cap \mathcal{N}(R)$$

is a linear subspace of $\mathbb{C}^n$ of dimension at least 1 by Exercise 2.3.12. Thus, there exists $\mathbf{n} \in \operatorname{span}\{V_{:,0}, \dots, V_{:,s}\} \cap \mathcal{N}(R)$ with $\|\mathbf{n}\|_2 = 1$.

Using the fact that $\mathbf{n} \in \mathcal{N}(R) \subset \mathcal{N}(B)$, and writing $\mathbf{n}$ as $\sum_{j \in [s+1]} \alpha_j V_{:,j}$ for some $\alpha_0, \dots, \alpha_s \in \mathbb{C}$ with $\|\boldsymbol{\alpha}\|_2 = 1$ (recall Theorem 2.3.9), we can now see that

$$\begin{aligned} \|A - B\|_{2\to 2} \;\geq\; \|(A-B)\mathbf{n}\|_2 \;=\; \|A\mathbf{n}\|_2 \;&=\; \left\| U\Sigma V^* \left(\sum_{j \in [s+1]} \alpha_j V_{:,j} \right) \right\|_2 \\ &= \sqrt{\sum_{j \in [s+1]} |\alpha_j|^2 \sigma_j^2(A)}. \end{aligned}$$

Recalling that $\|\boldsymbol{\alpha}\|_2 = 1$, we can now see that the expression above is minimized when $\alpha_j = 0$ for all $j < s$ so that $\alpha_s = 1$. Therefore,

$$\|A - B\|_{2\to 2} \geq \sigma_s(A).$$

We are now finished by (3.11). □

3.5 Solving Ill-Conditioned and Noisy Linear Systems

Noise, like death and taxes, is a sad and inescapable fact of life. Every single scientific, engineering, and computational problem you ever encounter will be littered with noise. In this section we will discuss the effect that noise has on the most fundamental linear algebra problem there is – solving a system of linear equations. In the process, we will present two classical strategies for mitigating the effects of noise.

To make our discussion more concrete, suppose that we are given access to a nonzero vector $\mathbf{b} \in \mathbb{C}^n$ and an invertible matrix $A \in \mathbb{C}^{n \times n}$. Our mission is to find a vector $\mathbf{x} \in \mathbb{C}^n$ such that $A\mathbf{x} = \mathbf{b}$. Not wanting to complete this task by hand (suppose $n > 100$) we instead opt to solve this simple linear system on a computer.[2] In the process of

[2] In fact, digital computers were invented to solve large systems of linear equations in the first place [2]. Somewhat interestingly, the inventor was an American mathematician of Bulgarian descent working at Iowa State University (then College), not at some ivy league university as one might mistakenly assume. Anyone working anywhere can have a huge impact![*]

[*] To anyone with connections to Iowa State who might be reading this footnote some day: oh yes, I am very aware that my motivational message is implicitly casting shade. Michigan State is obviously superior in every conceivable way. . . :).

loading/transferring/typing the matrix A and the vector $\mathbf{b}$ into the computer, though, something unpleasant happens. Though we somehow miraculously get A into computer memory with no errors, we do end up introducing small errors into $\mathbf{b}$ in the process, instead ending up with $\mathbf{b}' := \mathbf{b} + \boldsymbol{\epsilon}$ in its place. Note that this could happen in a thousand different ways. One obvious source of errors in our very simple example would be due to round-off error (i.e., due to the fact that digital computers can only represent a few of the most significant digits of any given number). Our only consolation in that case is that we can assume the "noise" vector $\boldsymbol{\epsilon} \in \mathbb{C}^n$ has a small norm compared to $\mathbf{b}$ (e.g., suppose we can tell that $\|\boldsymbol{\epsilon}\|_2 \leq 10^{-8}\|\mathbf{b}\|_2$).

Taking solace in the fact that $\boldsymbol{\epsilon} \in \mathbb{C}^n$ has a small norm, we go ahead and solve the linear system we currently have in computer memory,

$$A\mathbf{y} \;=\; \mathbf{b}' \;=\; \mathbf{b} + \boldsymbol{\epsilon}, \tag{3.12}$$

for $\mathbf{y} \in \mathbb{C}^n$ *instead* of solving the original equation $A\mathbf{x} = \mathbf{b}$ that we actually care about. After solving for $\mathbf{y}$ we then ask ourselves how close it is the true solution $\mathbf{x}$ we want. Doing a rough calculation of our relative error we see that

$$\begin{aligned}
\frac{\|\mathbf{x} - \mathbf{y}\|_2}{\|\mathbf{x}\|_2} &= \frac{\|A^{-1}\mathbf{b} - A^{-1}(\mathbf{b} + \boldsymbol{\epsilon})\|_2}{\|\mathbf{x}\|_2} = \frac{\|A^{-1}\boldsymbol{\epsilon}\|_2}{\|\mathbf{x}\|_2} \leq \frac{\|A^{-1}\|_{2\to2}\|\boldsymbol{\epsilon}\|_2}{\|\mathbf{x}\|_2} \\
&\leq \frac{\|A^{-1}\|_{2\to2}\left(10^{-8}\|\mathbf{b}\|_2\right)}{\|\mathbf{x}\|_2} = \frac{\|A^{-1}\|_{2\to2}\left(10^{-8}\|A\mathbf{x}\|_2\right)}{\|\mathbf{x}\|_2} \\
&\leq \frac{\|A^{-1}\|_{2\to2}\left(10^{-8}\|A\|_{2\to2}\|\mathbf{x}\|_2\right)}{\|\mathbf{x}\|_2} = 10^{-8}\left(\|A^{-1}\|_{2\to2}\|A\|_{2\to2}\right) \\
&= 10^{-8}\left(\frac{\sigma_0(A)}{\sigma_{n-1}(A)}\right).
\end{aligned} \tag{3.13}$$

Hence, our relative error will be small if $\frac{\sigma_0(A)}{\sigma_{n-1}(A)}$ is not too large. If this quantity is larger than 10^8, however, all bets are off – the relative error could be huge, and our approximate solution $\mathbf{y}$ effectively worthless.

Exercise 3.5.1. *Suppose that* $\|\boldsymbol{\epsilon}\|_2 = 10^{-8}\|\mathbf{b}\|_2$ *in* (3.13). *Show that in fact* $\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2} = 10^{-8}\left(\frac{\sigma_0(A)}{\sigma_{n-1}(A)}\right)$ *is achieved for a particular worst-case* $\mathbf{x}$ *and* $\boldsymbol{\epsilon}$*. What are they?*

Given its importance in applications, the quantity $\frac{\sigma_0(A)}{\sigma_{n-1}(A)}$ has a special name.

Definition 3.5.1 (Condition Number)**.** *Let* $A \in \mathbb{C}^{m\times n}$ *have rank* r*. The condition number of* A *is* $\kappa(A) := \left\|A^\dagger\right\|_{2\to2}\|A\|_{2\to2} = \frac{\sigma_0(A)}{\sigma_{r-1}(A)} \geq 1$.

Exercise 3.5.2. *Let* $U \in \mathbb{C}^{n\times n}$ *be a unitary matrix. Show that* $\kappa(U) = 1$.

Exercise 3.5.3. *Let* $A, B \in \mathbb{C}^{n\times n}$ *both be full rank. Show that* $\kappa(AB) \leq \kappa(A)\kappa(B)$.

Exercise 3.5.4. *Let $A \in \mathbb{C}^{m\times n}$ and $\alpha \in \mathbb{C} \setminus \{0\}$. Show that $\kappa(\alpha A) = \kappa(A)$.*

Exercise 3.5.5. *Let $A \in \mathbb{C}^{m\times n}$. Show that $\kappa\left(A^\dagger\right) = \kappa(A)$.*

Exercise 3.5.6. *Let $A_s \in \mathbb{C}^{m\times n}$ be the best rank-s approximation to $A \in \mathbb{C}^{m\times n}$ with respect the Frobenius and (ℓ^2, ℓ^2)-operator norms as per (3.9). Prove that $\kappa(A_s) \leq \kappa(A)$.*

Generally it indeed turns out to be true that the larger the condition number of A is, the larger the relative error $\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2}$ in (3.13) will end up being. For this reason matrices A with large condition numbers are said to be **ill-conditioned**, and specialized techniques are often used to try to invert them more accurately. We will now briefly discuss two such techniques.

3.5.1 Improving Conditioning by SVD Truncation

Let $A \in \mathbb{C}^{n\times n}$ be the full rank matrix in (3.12), and let $A_s \in \mathbb{C}^{n\times n}$ be its best rank-s approximation as per (3.9). As shown in Exercise 3.5.6, the condition number of A_s will always be less than or equal to the condition number of A. This suggests a relatively obvious strategy to combat poor conditioning. Instead of solving (3.12) directly by inverting A, we can instead solve the related least-squares problem

$$\mathbf{y} = \arg\min_{\mathbf{z}\in\mathbb{C}^n} \left\|A_s\mathbf{z} - \mathbf{b}'\right\|_2^2 \tag{3.14}$$

using the better-conditioned and lower-rank $A_s \in \mathbb{C}^{n\times n}$ in place of A. Recalling Sections 2.6.2 and 3.2, we can see that $\mathbf{y} = (A_s)^\dagger\mathbf{b}' = (A_s)^\dagger(\mathbf{b} + \boldsymbol{\epsilon})$ will solve (3.14). We can now ask ourselves what the relative error between this new $\mathbf{y}$ and the true solution $\mathbf{x} = A^{-1}\mathbf{b}$ will be in this case.

To help bound the relative error between $\mathbf{x} = A^{-1}\mathbf{b}$ and $\mathbf{y} = (A_s)^\dagger(\mathbf{b} + \boldsymbol{\epsilon})$ it will be useful to have a compact SVD of both A_s and $(A_s)^\dagger$. Toward that end, suppose that $A = U\Sigma V^*$ is a full SVD of the matrix A. From (3.9) we can then see that

$$A_s = U_s \begin{pmatrix} \sigma_0(A) & & \\ & \ddots & \\ & & \sigma_{s-1}(A) \end{pmatrix} (V_s)^*,$$

where $U_s, V_s \in \mathbb{C}^{n\times s}$ contain the first s orthonormal columns of $U, V \in \mathbb{C}^{n\times n}$, respectively. The pseudoinverse of A_s is therefore

$$(A_s)^\dagger = V_s \begin{pmatrix} 1/\sigma_0(A) & & \\ & \ddots & \\ & & 1/\sigma_{s-1}(A) \end{pmatrix} (U_s)^*. \tag{3.15}$$

Finally, using (3.15) we can further see, e.g., that

$$\begin{aligned}
(A_s)^\dagger A &= V_s \begin{pmatrix} 1/\sigma_0(A) & & \\ & \ddots & \\ & & 1/\sigma_{s-1}(A) \end{pmatrix} (U_s)^* U \begin{pmatrix} \sigma_0(A) & & \\ & \ddots & \\ & & \sigma_{n-1}(A) \end{pmatrix} V^* \\
&= V_s \begin{pmatrix} 1/\sigma_0(A) & & \\ & \ddots & \\ & & 1/\sigma_{s-1}(A) \end{pmatrix} (I_s \mid 0_{s\times n-s}) \begin{pmatrix} \sigma_0(A) & & \\ & \ddots & \\ & & \sigma_{n-1}(A) \end{pmatrix} V^* \\
&= V_s \begin{pmatrix} 1/\sigma_0(A) & & \\ & \ddots & \\ & & 1/\sigma_{s-1}(A) \end{pmatrix} \begin{pmatrix} \sigma_0(A) & & & 0 & \cdots & 0 \\ & \ddots & & 0 & \cdots & 0 \\ & & \sigma_{s-1}(A) & 0 & \cdots & 0 \end{pmatrix} V^* \\
&= V_s \left(I_s \mid 0_{s\times n-s}\right) V^* \;=\; V_s (V_s)^* \;=\; P_{\mathcal{C}((A_s)^*)}. \qquad (3.16)
\end{aligned}$$

Exercise 3.5.7. *Let $A \in \mathbb{C}^{n\times n}$ be full rank. Show that $A(A_s)^\dagger = P_{\mathcal{C}(A_s)}$.*

Using (3.16) to help bound the relative error between $\mathbf{x} = A^{-1}\mathbf{b}$ and $\mathbf{y} = (A_s)^\dagger(\mathbf{b}+\boldsymbol{\epsilon})$ we have that

$$\begin{aligned}
\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2} &= \frac{\|A^{-1}\mathbf{b} - (A_s)^\dagger(\mathbf{b}+\boldsymbol{\epsilon})\|_2}{\|\mathbf{x}\|_2} \leq \frac{\|(A_s)^\dagger\boldsymbol{\epsilon}\|_2}{\|\mathbf{x}\|_2} + \frac{\|\left((A_s)^\dagger - A^{-1}\right)\mathbf{b}\|_2}{\|\mathbf{x}\|_2} \\
&\leq \frac{\|(A_s)^\dagger\|_{2\to 2}\|\boldsymbol{\epsilon}\|_2}{\|\mathbf{x}\|_2} + \frac{\|\left((A_s)^\dagger A - I_n\right)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\
&\leq \frac{\|(A_s)^\dagger\|_{2\to 2}\cdot 10^{-8}\cdot\|\mathbf{b}\|_2}{\|\mathbf{x}\|_2} + \frac{\|\left(P_{\mathcal{C}((A_s)^*)} - I_n\right)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \qquad (3.17) \\
&\leq 10^{-8}\left(\frac{\sigma_0(A)}{\sigma_{s-1}(A)}\right) + \left\|P_{\mathcal{C}((A_s)^*)^\perp}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right)\right\|_2.
\end{aligned}$$

Suppose that $\sigma_{s-1}(A) > \sigma_{n-1}(A)$. Comparing (3.17) with (3.13) in this case we can see that (3.17) provides a smaller upper bound on the relative error $\|\mathbf{x}-\mathbf{y}\|_2/\|\mathbf{x}\|_2$ than (3.13) does whenever, e.g., $\mathbf{x} \in \mathcal{C}\left((A_s)^*\right)$. In particular, Exercise 3.5.1 gives an example where the relative error provided by (3.17) is a strict improvement over the relative error provided by (3.13) (check this!). More generally, since $\kappa(A_s) = \left(\frac{\sigma_0(A)}{\sigma_{s-1}(A)}\right) \leq \left(\frac{\sigma_0(A)}{\sigma_{n-1}(A)}\right) = \kappa(A)$ holds for all $s \in [n+1]\setminus\{0\}$, using $\mathbf{y} = (A_s)^\dagger(\mathbf{b}+\boldsymbol{\epsilon})$ will often lead to a better relative error than using $\mathbf{y} = A^{-1}(\mathbf{b}+\boldsymbol{\epsilon})$ when s is chosen so that

$$\left\|P_{\mathcal{C}((A_s)^*)^\perp}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right)\right\|_2 < \left(\kappa(A) - \kappa(A_s)\right)\frac{\|\boldsymbol{\epsilon}\|_2}{\|\mathbf{b}\|_2}.$$

We will next briefly discuss a different strategy for inverting ill-conditioned systems which doesn't require us to entirely discard all of $P_{\mathcal{C}((A_s)^*)^\perp}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right)$.

3.5.2 Tikhonov Regularization

Let $A \in \mathbb{C}^{n \times n}$ again be the full rank matrix in (3.12). This time, though, suppose that we pick a positive real $\alpha \in (0, \infty)$ and then solve the modified least-squares problem

$$\mathbf{y} = \arg\min_{\mathbf{z} \in \mathbb{C}^n} \|A\mathbf{z} - \mathbf{b}'\|_2^2 + \alpha \|\mathbf{z}\|_2^2 \tag{3.18}$$

instead of solving (3.12). As $\alpha \to 0$ we can see that the solution of (3.18) will converge to the solution of (3.12), so we expect that using a small α should still give us about the same answer. As we shall see below, though, there are benefits to solving (3.18) with $\alpha > 0$ instead of solving (3.12) when $\kappa(A)$ is very large.

Solving (3.18) in order to estimate $\mathbf{x} = A^{-1}\mathbf{b}$ is known as *Tikhonov regularization*, and the number α above is referred to as the *regularization parameter*. Note that (3.18) has less overhead to set up and start solving than (3.14) does because (3.18) doesn't require us to compute A_s. For this reason solving (3.18) is often preferable to solving (3.14) when A is very large and we don't expect to reuse it many times.

With a little bit of work one can see that the unique solution to (3.18) is given by $\mathbf{y} = (\alpha I_n + A^*A)^{-1} A^*\mathbf{b}'$ (see, e.g., [32, Theorem 5.9]). The following exercises will help us understand this closed-form solution in more detail.

Exercise 3.5.8. *Let $\alpha \in (0, \infty)$ and $A \in \mathbb{C}^{n \times n}$. Show that the matrix $\alpha I_n + A^*A$ is positive definite (and therefore invertible). Furthermore, show that $\sigma_j(\alpha I_n + A^*A) = \alpha + \sigma_j^2(A) > 0$ for all $j \in [n]$.*

Exercise 3.5.9. *Let $\alpha \in (0, \infty)$ and $A \in \mathbb{C}^{n \times n}$. Show that $\forall \ell \in [n]$ $\exists j \in [n]$ such that*

$$\sigma_\ell\left((\alpha I_n + A^*A)^{-1} A^*\right) = \frac{\sigma_j(A)}{\alpha + \sigma_j^2(A)}.$$

Exercise 3.5.10. *Let $\alpha \in (0, \infty)$ and $A \in \mathbb{C}^{n \times n}$. Prove that*

$$\frac{x}{x^2 + \alpha} \leq \begin{cases} 1 & \text{if } x \leq \alpha \\ \frac{1}{\alpha} & \text{if } x > \alpha \end{cases}.$$

Conclude that $\left\|(\alpha I_n + A^*A)^{-1} A^*\right\|_{2\to 2} \leq \max\left\{\frac{1}{\alpha}, 1\right\}$.

Exercise 3.5.11. *Let $\alpha \in (0, \infty)$ and $A \in \mathbb{C}^{n \times n}$. Show that*

$$\sigma_j\left(I_n - (\alpha I_n + A^*A)^{-1} A^*A\right) = \frac{\alpha}{\sigma_{n-j-1}^2(A) + \alpha}$$

holds for all $j \in [n]$.

We can now ask ourselves what the relative error between this new $\mathbf{y} = \left(\alpha I_n + A^*A\right)^{-1} A^*\mathbf{b}'$ and the true solution $\mathbf{x} = A^{-1}\mathbf{b}$ will be in this case. Bounding the relative error using Exercises 3.5.10 and 3.5.11 together with our assumption that $\|\boldsymbol{\epsilon}\|_2 \leq 10^{-8}\|\mathbf{b}\|_2$ we see that

$$\begin{aligned} \frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2} &= \frac{\|A^{-1}\mathbf{b} - \left(\alpha I_n + A^*A\right)^{-1} A^*(\mathbf{b}+\boldsymbol{\epsilon})\|_2}{\|\mathbf{x}\|_2} \\ &\leq \frac{\left\|\left(\alpha I_n + A^*A\right)^{-1} A^*\boldsymbol{\epsilon}\right\|_2}{\|\mathbf{x}\|_2} + \frac{\left\|A^{-1}\mathbf{b} - \left(\alpha I_n + A^*A\right)^{-1} A^*\mathbf{b}\right\|_2}{\|\mathbf{x}\|_2} \\ &\leq \frac{\max\left\{\frac{1}{\alpha}, 1\right\} \cdot \|\mathbf{b}\|_2 \cdot 10^{-8}}{\|\mathbf{x}\|_2} + \frac{\left\|\left(I_n - \left(\alpha I_n + A^*A\right)^{-1} A^*A\right)\mathbf{x}\right\|_2}{\|\mathbf{x}\|_2} \\ &= \frac{\max\left\{\frac{1}{\alpha}, 1\right\} \cdot \|\mathbf{b}\|_2 \cdot 10^{-8}}{\|\mathbf{x}\|_2} + \left\|\underbrace{\left(I_n - \left(\alpha I_n + A^*A\right)^{-1} A^*A\right)}_{E_\alpha :=} \frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right\|_2 \qquad (3.19) \\ &\leq \max\left\{\frac{1}{\alpha}, 1\right\} \cdot \sigma_0(A) \cdot 10^{-8} + \frac{\alpha}{\sigma_{n-1}^2(A) + \alpha}. \end{aligned}$$

Suppose that $\alpha > \sigma_{n-1}(A)$. Comparing (3.19) with (3.13) in this case we can see that (3.19) provides a better upper bound on the relative error $\|\mathbf{x}-\mathbf{y}\|_2/\|\mathbf{x}\|_2$ than (3.13) does whenever α is chosen so that

$$\frac{\|E_\alpha \mathbf{x}\|_2}{\sigma_0(A)\|\mathbf{x}\|_2} < \left(\frac{1}{\sigma_{n-1}(A)} - \frac{1}{\alpha}\right)\frac{\|\boldsymbol{\epsilon}\|_2}{\|\mathbf{b}\|_2}.$$

Indeed, Tikhonov regularization generally reduces relative errors in practice as long as some care is taken in properly selecting the regularization parameter α. And, of course, that's the trick: if we truly only have access to $\mathbf{b}'$ and A then we really don't have enough information to select an optimal α (or s in Section 3.5.1). The best one can do in such cases (without the aid of additional side information at least) is to choose α (or s) so that $\frac{\sigma_0(A)}{\alpha}$ (or $\kappa(A_s)$) is not "too large".

3.6 Linear Least-Squares Regression

Suppose that we have $p \in \mathbb{N}$ evaluations of a (potentially) unknown function $f : \mathbb{C}^n \to \mathbb{C}^m$. Here the reader should imagine that p (especially) as well as both n and m are potentially very large. As a result, we'd like to find a low rank (and therefore compressible by Section 2.5) matrix $A \in \mathbb{C}^{m\times n}$, and a shift vector $\mathbf{b} \in \mathbb{C}^m$, so that $f(\mathbf{x}) \approx A\mathbf{x} + \mathbf{b}\ \forall \mathbf{x} \in \mathbb{C}^n$. Of course, this is generally impossible since we only have p evaluations of f in the form of a dataset

$$\mathcal{D} := \left\{(\mathbf{x}_j, f(\mathbf{x}_j))\right\}_{j\in[p]} \ = \ \left\{(\mathbf{x}_j, \mathbf{y}_j)\right\}_{j\in[p]} \subset \mathbb{C}^n \times \mathbb{C}^m,$$

where $\mathbf{y}_j := f(\mathbf{x}_j)$ for all $j \in [p]$. Thus, we will settle for finding a rank at most s matrix $A \in \mathbb{C}^{m \times n}$ and $\mathbf{b} \in \mathbb{C}^m$ that matches f on the available data $\mathcal{D}$ as well as possible. More specifically, we seek a rank s matrix $A \in \mathbb{C}^{m \times n}$ and $\mathbf{b} \in \mathbb{C}^m$ satisfying

$$\frac{1}{2} \sum_{j \in [p]} \|A\mathbf{x}_j + \mathbf{b} - \mathbf{y}_j\|_2^2 \; = \min_{\substack{B \in \mathbb{C}^{m \times n}, \; \mathbf{d} \in \mathbb{C}^m \\ \operatorname{rank}(B) \, \leq \, s}} \frac{1}{2} \sum_{j \in [p]} \|B\mathbf{x}_j + \mathbf{d} - \mathbf{y}_j\|_2^2. \tag{3.20}$$

Our first step will be to show that centering the evaluation points, $\{\mathbf{x}_j\}_{j \in [p]}$, allows us to decouple the choice of an optimal shift $\mathbf{b}$ from the choice of an optimal rank s matrix $A \in \mathbb{C}^{m \times n}$. Once this decoupling is achieved we'll see that a straightforward solution to our problem already exists.

3.6.1 Centering, and the Optimal Shift $\mathbf{b} \in \mathbb{C}^m$

Let $\mathbf{c} := \frac{1}{p} \sum_{j \in [p]} \mathbf{x}_j$. Instead of directly approximating $f : \mathbb{C}^n \to \mathbb{C}^m$ using the available data $\mathcal{D}$, we will instead approximate the new function $g : \mathbb{C}^n \to \mathbb{C}^m$ defined by $g(\mathbf{x}) := f(\mathbf{x} + \mathbf{c})$ (so that $f(\mathbf{x}) = g(\mathbf{x} - \mathbf{c})$ and $g(\mathbf{x}_j - \mathbf{c}) = f(\mathbf{x}_j) = \mathbf{y}_j \; \forall j \in [p]$) using the centered data

$$\mathcal{D}' := \{(\mathbf{x}_j - \mathbf{c}, \mathbf{y}_j)\}_{j \in [p]} \; = \; \left\{\left(\mathbf{x}'_j, \mathbf{y}_j\right)\right\}_{j \in [p]},$$

where $\mathbf{x}'_j := \mathbf{x}_j - \mathbf{c} \; \forall j \in [p]$. Toward re-expressing (3.20) in terms of our centered data $\mathcal{D}'$ let $q : \mathbb{C}^{m \times n} \times \mathbb{C}^m \to \mathbb{R}^+$ be defined by

$$q\left(B, \mathbf{d}\right) \; := \; \frac{1}{2} \sum_{j \in [p]} \left\|B\mathbf{x}'_j + \mathbf{d} - \mathbf{y}_j\right\|_2^2.$$

We can now see that solving (3.20) is equivalent to computing

$$(A, \mathbf{b}) = \arg\min_{\substack{B \in \mathbb{C}^{m \times n}, \; \mathbf{d} \in \mathbb{C}^m \\ \operatorname{rank}(B) \, \leq \, s}} q(B, \mathbf{d}) \tag{3.21}$$

up to an additional shift of $\mathbf{b}$ by $-A\mathbf{c}$.

Exercise 3.6.1. *Let $\{\mathbf{x}'_j\}_{j \in [p]}$ be defined as above. Verify that $\sum_{j \in [p]} \mathbf{x}'_j = \mathbf{0}$.*

The next lemma tells us that $\mathbf{b} = \frac{1}{p} \sum_{j \in [p]} \mathbf{y}_j$ is always an optimal choice of the shift in (3.21), completely independent of the matrix A.

Lemma 3.6.1. *Let $q : \mathbb{C}^{m \times n} \times \mathbb{C}^m \to \mathbb{R}^+$ be as in (3.21). Then*

$$q\left(B, \; \frac{1}{p} \sum_{j \in [p]} \mathbf{y}_j\right) \; \leq \; q\left(B, \mathbf{d}\right)$$

holds for all $B \in \mathbb{C}^{m \times n}$ and $\mathbf{d} \in \mathbb{C}^m$.

Proof. Fix $B \in \mathbb{C}^{m \times n}$ and note that

$$\begin{aligned} q\left(B,\mathbf{d}\right) &= \frac{1}{2}\sum_{j\in[p]}\sum_{k\in[m]}\left|\left(B\mathbf{x}'_j\right)_k + d_k - (\mathbf{y}_j)_k\right|^2 \\ &= \frac{1}{2}\sum_{j\in[p]}\sum_{k\in[m]}\Bigg[\left(\mathrm{Re}\left(\left(B\mathbf{x}'_j\right)_k\right) + \mathrm{Re}\left(d_k\right) - \mathrm{Re}\left((\mathbf{y}_j)_k\right)\right)^2 \\ &\qquad\qquad + \left(\mathrm{Im}\left(\left(B\mathbf{x}'_j\right)_k\right) + \mathrm{Im}\left(d_k\right) - \mathrm{Im}\left((\mathbf{y}_j)_k\right)\right)^2\Bigg]. \end{aligned}$$

From above we can see that the function $q(B,\cdot) : \mathbb{C}^m \to \mathbb{R}^+$ ultimately only depends on the $2m$ real and imaginary parts of it's m complex input variables (i.e., the d_k values above). As a result, $q(B,\cdot)$ can also be treated as a function from $\mathbb{R}^{2m}$ into $\mathbb{R}^+$.

Let $\ell \in [m]$. Recalling multivariable calculus (see, e.g., [33, Chapter 3]), we now know that any critical point $\mathbf{a} \in \mathbb{C}^m$ of $q(B,\cdot) : \mathbb{C}^m \to \mathbb{R}^+$ will satisfy

$$\begin{aligned} 0 &= \frac{\partial q}{\partial \mathrm{Re}\left(a_\ell\right)} = \sum_{j\in[p]}\mathrm{Re}\left(\left(B\mathbf{x}'_j\right)_\ell\right) + \mathrm{Re}\left(a_\ell\right) - \mathrm{Re}\left((\mathbf{y}_j)_\ell\right) \\ &= \mathrm{Re}\left(\left(B\sum_{j\in[p]}\mathbf{x}'_j\right)_\ell\right) + p\mathrm{Re}\left(a_\ell\right) - \mathrm{Re}\left(\sum_{j\in[p]}(\mathbf{y}_j)_\ell\right) \qquad (3.22) \\ &= p\mathrm{Re}\left(a_\ell\right) - \mathrm{Re}\left(\sum_{j\in[p]}(\mathbf{y}_j)_\ell\right), \end{aligned}$$

where the last equality holds by Exercise 3.6.1. Rearranging (3.22) we can now see that $\mathrm{Re}\left(a_\ell\right) = \mathrm{Re}\left(\frac{1}{p}\sum_{j\in[p]}(\mathbf{y}_j)_\ell\right)$ must hold. A completely analogous argument similarly shows that $\mathrm{Im}\left(a_\ell\right) = \mathrm{Im}\left(\frac{1}{p}\sum_{j\in[p]}(\mathbf{y}_j)_\ell\right)$. As a consequence, we have that $a_\ell = \frac{1}{p}\sum_{j\in[p]}(\mathbf{y}_j)_\ell = \left(\frac{1}{p}\sum_{j\in[p]}\mathbf{y}_j\right)_\ell$ holds for all $\ell \in [m]$.

The result now follows after observing that $q(B,\cdot)$ is both always nonnegative-valued, and quadratic in all $2m$ of its input variables' real/imaginary parts. Hence, its single critical point $\mathbf{a} = \frac{1}{p}\sum_{j\in[p]}\mathbf{y}_j$ will be a global minimizer. Finally, we note that this minimizer is completely independent of the matrix B. □

Let $\mathbf{y}'_j := \mathbf{y}_j - \left(\frac{1}{p}\sum_{k\in[p]}\mathbf{y}_k\right)$ for all $j \in [p]$, and then define $X \in \mathbb{C}^{n\times p}$ and $Y \in \mathbb{C}^{m\times p}$ to be such that $X_{:,j} := \mathbf{x}'_j$ and $Y_{:,j} := \mathbf{y}'_j$ hold for all $j \in [p]$. We can now see that computing

(3.21) is equivalent to computing

$$\begin{aligned} A &= \arg\min_{\substack{B \in \mathbb{C}^{m\times n} \text{ s.t.} \\ \operatorname{rank}(B) \,\leq\, s}} \frac{1}{2} \sum_{j\in[p]} \left\| B\mathbf{x}'_j - \mathbf{y}'_j \right\|_2^2 \;=\; \arg\min_{\substack{B \in \mathbb{C}^{m\times n} \text{ s.t.} \\ \operatorname{rank}(B) \,\leq\, s}} \frac{1}{2} \left\| BX - Y \right\|_{\mathrm{F}}^2 \\ &= \arg\min_{\substack{B \in \mathbb{C}^{m\times n} \text{ s.t.} \\ \operatorname{rank}(B) \,\leq\, s}} \frac{1}{2} \left\| X^*B^* - Y^* \right\|_{\mathrm{F}}^2 \\ &= \arg\min_{\substack{B \in \mathbb{C}^{m\times n} \text{ s.t.} \\ \operatorname{rank}(B) \,\leq\, s}} \frac{1}{2} \left\| X^*B^* - P_{\mathcal{C}(X^*)}Y^* \right\|_{\mathrm{F}}^2 \\ &= \arg\min_{\substack{B \in \mathbb{C}^{m\times n} \text{ s.t.} \\ \operatorname{rank}(B) \,\leq\, s}} \frac{1}{2} \left\| BX - YP_{\mathcal{C}(X^*)} \right\|_{\mathrm{F}}^2 . \end{aligned} \tag{3.23}$$

3.6.2 The Optimal Low-Rank Matrix $A \in \mathbb{C}^{m\times n}$

Assume for simplicity that $p > n$, and that $X \in \mathbb{C}^{n\times p}$ is full rank so that $\mathcal{C}(X) = \mathbb{C}^n$. Recalling Section 3.4.1 in the context of (3.23) we can now see that $AX = \left(YP_{\mathcal{C}(X^*)}\right)_s = \left(YP_{\mathcal{C}(X^*)}\right)_s P_{\mathcal{C}(X^*)}$ must hold. As a result we will have that

$$A = AXX^\dagger = \left(YP_{\mathcal{C}(X^*)}\right)_s X^\dagger$$

since, by Exercise 3.2.1 together with the assumption that $\mathcal{C}(X) = \mathbb{C}^n$, $XX^\dagger = P_{\mathcal{C}(X)} = I_n$.

Exercise 3.6.2. *Show that $\mathcal{N}(X) = \mathcal{C}(X^*)^\perp \subset \mathcal{N}\left(YP_{\mathcal{C}(X^*)}\right)$.*

Exercise 3.6.3. *Let $\{\mathbf{v}_j\}_{j\in[p]} \subset \mathbb{C}^p$ be right singular vectors of $YP_{\mathcal{C}(X^*)}$ ordered corresponding to their associated singular values. Prove that $\mathcal{C}(X^*)^\perp \subset$ span $\{\mathbf{v}_j\}_{j=\operatorname{rank}\left(YP_{\mathcal{C}(X^*)}\right)}^{p-1}$.*

Exercise 3.6.4. *Prove that the first* rank $\left(YP_{\mathcal{C}(X^*)}\right)$ *right singular vectors of any SVD of $YP_{\mathcal{C}(X^*)}$ belong to $C(X^*)$.*

Exercise 3.6.5. *Prove that $\left(YP_{\mathcal{C}(X^*)}\right)_s = \left(YP_{\mathcal{C}(X^*)}\right)_s P_{\mathcal{C}(X^*)}$ for all $s \in [\min\{m,p\}]$.*
HINT: Let $\{\mathbf{v}_j\}_{j\in[p]} \subset \mathbb{C}^p$ be right singular vectors of $YP_{\mathcal{C}(X^*)}$. Show that $\left(YP_{\mathcal{C}(X^*)}\right)_s \mathbf{v}_j = \left(YP_{\mathcal{C}(X^*)}\right)_s P_{\mathcal{C}(X^*)}\mathbf{v}_j$ for all $j \in [p]$, and then appeal to Exercise 2.6.10.

The following theorem follows from the discussion above.

Theorem 3.6.2 (Linear Least-Squares Regression)**.** *Given the data $\mathcal{D} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j\in[p]} \subset \mathbb{C}^n \times \mathbb{C}^m$ let $\mathbf{c_x} := \frac{1}{p}\sum_{j\in[p]} \mathbf{x}_j \in \mathbb{C}^n$, $\mathbf{c_y} := \frac{1}{p}\sum_{j\in[p]} \mathbf{y}_j \in \mathbb{C}^m$, and define $X \in \mathbb{C}^{n\times p}$ and $Y \in \mathbb{C}^{m\times p}$ to be such that $X_{:,j} := \mathbf{x}_j - \mathbf{c_x}$ and $Y_{:,j} := \mathbf{y}_j - \mathbf{c_y}$ for all $j \in [p]$. Furthermore, suppose that $p > n$ and that X is full rank. Choose $s \in \{1, \dots, \min\{m,n\}\}$ and set $A := \left(YP_{\mathcal{C}(X^*)}\right)_s X^\dagger$ and $\mathbf{b} := \mathbf{c_y} - A\mathbf{c_x}$. Then, this A and $\mathbf{b}$ satisfy* (3.20)*.*

We will now consider a couple special cases of Theorem 3.6.2 which are commonly used in applications.

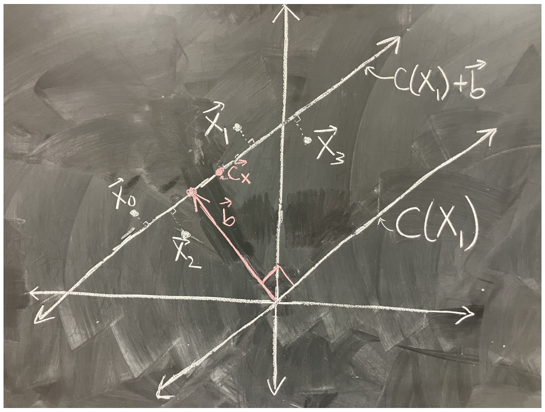

Figure 3.1: A pictorial representation of the best 1-dimensional least-squares line for four data points $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\} \subset \mathbb{R}^2$ (i.e., here $s = 1$). In this case the matrix $X = \begin{pmatrix} \mathbf{x}_0 - \mathbf{c}_\mathbf{x} & \mathbf{x}_1 - \mathbf{c}_\mathbf{x} & \mathbf{x}_2 - \mathbf{c}_\mathbf{x} & \mathbf{x}_3 - \mathbf{c}_\mathbf{x} \end{pmatrix} \in \mathbb{R}^{2\times 4}$, and $\mathcal{C}(X_1)$ is the linear subspace spanned by the principal left singular vector of X. The affine subspace $\mathcal{C}(X_1) + \mathbf{b}$ minimizes the sum of the squared distances to the four data points.

Example 3.6.3 (Point Cloud Compression)**.** *Suppose that we want to compress a large set of points $\{\mathbf{x}_j\}_{j\in[p]} \subset \mathbb{C}^n$ with $p > n$. Finding a rank $s \in \{1,\dots,n\}$ matrix $A \in \mathbb{C}^{n\times n}$ and shift $\mathbf{b} \in \mathbb{C}^n$ that compresses these points optimally in the least-squares sense is equivalent to solving* (3.20) *with $\mathbf{y}_j = \mathbf{x}_j$ for all $j \in [p]$. Let $\mathbf{c}_\mathbf{x} := \frac{1}{p}\sum_{j\in[p]} \mathbf{x}_j \in \mathbb{C}^n$ and define $X \in \mathbb{C}^{n\times p}$ to be such that $X_{:,j} := \mathbf{x}_j - \mathbf{c}_\mathbf{x}$ for all $j \in [p]$ (noting that $Y = X$ and $\mathbf{c}_\mathbf{y} = \mathbf{c}_\mathbf{x}$ will also hold in Theorem 3.6.2 in this case). Theorem 3.6.2 together with Exercises 3.6.7 and 3.6.6 now tell us that an optimal solution is given by $A = X_s X^\dagger = P_{\mathcal{C}(X_s)}$ and $\mathbf{b} = \left(I_n - P_{\mathcal{C}(X_s)}\right)\mathbf{c}_\mathbf{x} = P_{\mathcal{C}(X_s)^\perp}\mathbf{c}_\mathbf{x}$ provided that X is full rank.*

Simplifying even further, suppose that we want to compress $p = 4$ data points $\{\mathbf{x}_j\}_{j\in[3]} \subset \mathbb{R}^2$ using an $(s = 1)$-dimensional matrix $A \in \mathbb{C}^{2\times 2}$. In this case we can see that $\mathbf{c}_\mathbf{x} = \frac{1}{4}(\mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$, $X = \begin{pmatrix} \mathbf{x}_0 - \mathbf{c}_\mathbf{x} & \mathbf{x}_1 - \mathbf{c}_\mathbf{x} & \mathbf{x}_2 - \mathbf{c}_\mathbf{x} & \mathbf{x}_3 - \mathbf{c}_\mathbf{x} \end{pmatrix} \in \mathbb{R}^{2\times 4}$, $A = P_{\mathcal{C}(X_1)} \in \mathbb{R}^{2\times 2}$, and $\mathbf{b} = P_{\mathcal{C}(X_1)^\perp}\mathbf{c}_\mathbf{x} \in \mathbb{R}^2$. Theorem 3.6.2 guarantees that the resulting line yielded by A and $\mathbf{b}$, $\mathcal{C}(X_1) + \mathbf{b}$, obtains the minimal sum of the squared ℓ^2-distances to the four data points achievable by any one-dimensional affine subspace (see Figure 3.1).

Exercise 3.6.6. *Let $A \in \mathbb{C}^{m\times n}$ and $s \in \{1,\dots,\min\{m,n\}\}$. Show that $A_s A^\dagger = P_{\mathcal{C}(A_s)}$ and that $A^\dagger A_s = P_{\mathcal{C}(A_s^*)}$.*

Exercise 3.6.7. *Let $X \in \mathbb{C}^{n\times p}$ and $s \in \{1,\dots,\min\{n,p\}\}$. Show that $\left(X P_{\mathcal{C}(X^*)}\right)_s = X_s$.*

Exercise 3.6.8. *Let $X \in \mathbb{C}^{n\times p}$ and $s \in \{1,\dots,\min\{n,p\}\}$. Show that $P_{\mathcal{C}(X_s)} X = X_s$.*

Example 3.6.4 (Least-Squares Regression in the Plane). *Suppose that $\mathcal{D} = \{(x'_j, y'_j)\}_{j\in[p]} \subset \mathbb{C} \times \mathbb{C} \equiv \mathbb{C}^2$ (so that $n = m = 1$ in Theorem 3.6.2). In this case we want to find a best $m = A \in \mathbb{C}$ (a "1×1" matrix) and $b \in \mathbb{C}$ so that $mx'_j + b \approx y'_j$ for all $j \in [p]$ (i.e., $s = 1$). Continuing, we can further see that $c_x := \frac{1}{p}\sum_{j\in[p]} x'_j \in \mathbb{C}$ and $c_y := \frac{1}{p}\sum_{j\in[p]} y'_j \in \mathbb{C}$. Setting $x_j := x'_j - c_x$ and $y_j := y'_j - c_y$ for all $j \in [p]$, we have that $X \in \mathbb{C}^{1\times p}$ from Theorem 3.6.2 is $\mathbf{x}^T$ (the transpose of the vector $\mathbf{x} \in \mathbb{C}^p$ whose entries are the x_j). Similarly, $Y = \mathbf{y}^T \in \mathbb{C}^{1\times p}$. Finally, letting $\overline{\mathbf{x}} \in \mathbb{C}^p$ have entries $(\overline{x})_j := \overline{x_j}$ for all $j \in [p]$, we also find that $P_{\mathcal{C}(X^*)} = P_{\mathcal{C}(\overline{\mathbf{x}})} = \frac{\overline{\mathbf{x}}\,\overline{\mathbf{x}}^*}{\|\mathbf{x}\|_2^2} \in \mathbb{C}^{p\times p}$ and $X^\dagger = \frac{\overline{\mathbf{x}}}{\|\mathbf{x}\|_2^2} \in \mathbb{C}^p$ so that*

$$m \;=\; A \;=\; \left(YP_{\mathcal{C}(X^*)}\right)_1 X^\dagger \;=\; YP_{\mathcal{C}(X^*)}X^\dagger \;=\; \mathbf{y}^T\frac{\overline{\mathbf{x}}\,\overline{\mathbf{x}}^*}{\|\mathbf{x}\|_2^2}\frac{\overline{\mathbf{x}}}{\|\mathbf{x}\|_2^2} \;=\; \frac{\langle \mathbf{x}, \mathbf{y}\rangle}{\|\mathbf{x}\|_2^2} \in \mathbb{C}, \qquad (3.24)$$

and $\mathbf{b} = c_y - mc_x \in \mathbb{C}$.

For a concrete numerical example, consider the dataset $\mathcal{D} = \{(x'_j, y'_j)\}_{j\in[4]} = \{(1,6), (2,5), (3,7), (4,10)\} \subset \mathbb{R} \times \mathbb{R} \equiv \mathbb{R}^2$. We want to find a line $\ell(x) = mx + b$ minimizing

$$\frac{1}{2}\sum_{j\in[4]} \left(\ell\left(x'_j\right) - y'_j\right)^2 \;=\; \frac{1}{2}\left((\ell(1)-6)^2 + (\ell(2)-5)^2 + (\ell(3)-7)^2 + (\ell(4)-10)^2\right).$$

Proceeding as above we can see that $c_x = \frac{1}{4}(1+2+3+4) = \frac{5}{2}$ and $c_y = \frac{1}{4}(6+5+7+10) = 7$. As a consequence we have that

$$\mathbf{x} = \begin{pmatrix}1\\2\\3\\4\end{pmatrix} - \frac{5}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}-3/2\\-1/2\\1/2\\3/2\end{pmatrix}, \text{ and } \mathbf{y} = \begin{pmatrix}6\\5\\7\\10\end{pmatrix} - 7\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}-1\\-2\\0\\3\end{pmatrix}$$

in this case. Using (3.24) we can now see that

$$m \;=\; \frac{\langle \mathbf{x}, \mathbf{y}\rangle}{\|\mathbf{x}\|_2^2} \;=\; \frac{3/2+1+9/2}{9/4+1/4+1/4+9/4} \;=\; \frac{7}{5} \;=\; 1.4$$

so that $b = c_y - mc_x = 7 - (7/5)(5/2) = 7/2$. Hence, $\ell(x) = 1.4x + 3.5$ is the best-fit line (see Figure 3.2).

3.7 Discrete Convolution and Fourier Transform Matrices

We begin this section by defining a general class of matrices which are important in many applications including, e.g., as the weight matrices used in a special type of neural network layer known as a "convolutional" neural network layer (recall Definition 1.2.4). As will be clear soon, one advantage of this type of matrix is that it's defined with many fewer parameters than a generic matrix requires.

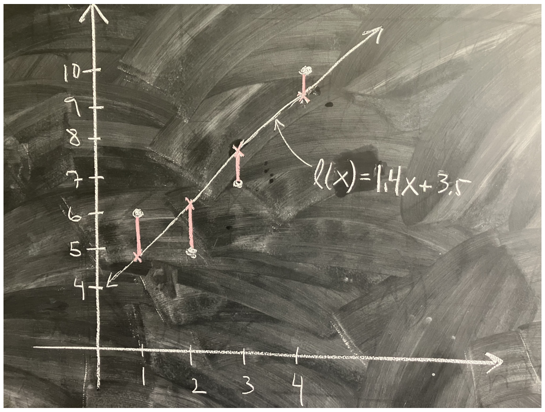

Figure 3.2: A pictorial representation of the best-fit least-squares line $\ell(x) = 1.4x + 3.5$ for the data $\mathcal{D} = \{(1,6),\ (2,5),\ (3,7),\ (4,10)\} \subset \mathbb{R} \times \mathbb{R} \equiv \mathbb{R}^2$ from Example 3.6.4.

Definition 3.7.1 (Toeplitz Matrix)**.** *The* ***Toeplitz matrix*** $A \in \mathbb{C}^{m\times n}$ ***generated by the vector*** $\mathbf{a} \in \mathbb{C}^{m+n-1}$ *is the* $\mathbb{C}^{m\times n}$ *matrix with entries given by*

$$A_{j,k} := a_{(m-1)+k-j}$$

for all $j \in [m], k \in [n]$*. We will also denote this matrix by* $A = \mathrm{Toep}_{m,n}(\mathbf{a})$*. More generally, we will say that a matrix* $A \in \mathbb{C}^{m\times n}$ *is* **Toeplitz** *if there exists a vector* $\mathbf{a} \in \mathbb{C}^{m+n-1}$ *such that* $A = \mathrm{Toep}_{m,n}(\mathbf{a})$*. We will also define* $\mathrm{Toep}_n(\mathbf{a}) := \mathrm{Toep}_{n,n}(\mathbf{a})$ *in the case of square matrices.*

Example 3.7.2. *The Toeplitz matrix* $A \in \mathbb{C}^{3\times 4}$ *generated by* $\mathbf{a} \in \mathbb{C}^6$ *is*

$$\mathrm{Toep}_{3,4}(\mathbf{a}) = \begin{pmatrix} a_2 & a_3 & a_4 & a_5 \\ a_1 & a_2 & a_3 & a_4 \\ a_0 & a_1 & a_2 & a_3 \end{pmatrix}.$$

The Toeplitz matrix $A \in \mathbb{C}^{4\times 3}$ *generated by* $\mathbf{a} \in \mathbb{C}^6$ *is*

$$\mathrm{Toep}_{4,3}(\mathbf{a}) = \begin{pmatrix} a_3 & a_4 & a_5 \\ a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 \\ a_0 & a_1 & a_2 \end{pmatrix}.$$

Note that the entries of **a** *appear along the bottom row, and then up the rightmost row, of the Toeplitz matrix it generates in a "backwards-L" shape (displayed in* blue *above). The rest of the Toeplitz matrix is then determined by its being constant along all of its diagonals.*

Exercise 3.7.1. *Show that* $A \in \mathbb{C}^{m\times n}$ *is Toeplitz if and only if* $A_{j,k} = A_{j+1,k+1}$ *holds for all* $j \in [m-1]$ *and* $k \in [n-1]$*.*

Exercise 3.7.2. *Show that A is Toeplitz if and only if A^* is Toeplitz.*

Exercise 3.7.3. *Given $\mathbf{a} \in \mathbb{C}^n$ let $\mathrm{Reverse}(\mathbf{a}) \in \mathbb{C}^n$ be the vector with entries given by*

$$\left(\mathrm{Reverse}(\mathbf{a})\right)_j = \overline{a_{n-1-j}}.$$

Show that if $A \in \mathbb{C}^{m \times n}$ is the Toeplitz matrix generated by $\mathbf{a} \in \mathbb{C}^{m+n-1}$, then A^* is the Toeplitz matrix generated by $\mathrm{Reverse}(\mathbf{a})$.

Definition 3.7.3 (Convolutional Layer of Neurons)**.** *A* **Convolutional Layer of Neurons** *$\ell : \mathbb{R}^N \to \mathbb{R}^d$ is a layer of neurons (recall Definition 1.2.4), $\sigma\left(W\mathbf{x} + \mathbf{b}\right)$, where the weight matrix $W \in \mathbb{R}^{d \times N}$ is Toeplitz.*

We can now see that an $m \times n$ Toeplitz matrix is entirely defined using only $m+n-1 < mn$ parameters. This can have potential benefits during, e.g., NN training. Of more immediate interest in this section, however, is that these matrices can also have runtime advantages as linear functions when coupled with Discrete Fourier Transform techniques. This will be discussed in Sections 3.7.2 and 3.8. Before we can understand how these computational advantages appear, however, we first have to discuss a special type of square Toeplitz matrices known as "circulant matrices".

3.7.1 Circulant and Toeplitz Matrices

As we will see later in Section 3.7.2, the following special class of square Toeplitz matrices is crucial to realizing fast matrix-vector multiplication algorithms for more arbitrary Toeplitz matrices.

Definition 3.7.4 (Circulant Matrix)**.** *The* ***circulant matrix generated by a vector*** *$\mathbf{v} \in \mathbb{C}^n$ is the matrix $\mathrm{circ}(\mathbf{v}) \in \mathbb{C}^{n \times n}$ defined by*

$$\left(\mathrm{circ}(\mathbf{v})\right)_{j,k} = v_{(j-k) \bmod n}.$$

We will say that a matrix $A \in \mathbb{C}^{n \times n}$ is **circulant** *if there exists a vector $\mathbf{v} \in \mathbb{C}^n$ such that $A = \mathrm{circ}(\mathbf{v})$. Recall that "$j \bmod n$" is defined for all $j \in \mathbb{Z}$ and $n \in \mathbb{N}$ to be the single element contained in the set $\{j + kn \mid k \in \mathbb{Z}\} \cap [n]$ (or, equivalently, to be the unique value $r \in [n] = \{0, 1, \dots, n-1\}$ such that $\exists k \in \mathbb{Z}$ satisfying $j = r + kn$).*

Example 3.7.5. *The circulant matrix $A \in \mathbb{C}^{4 \times 4}$ generated by $\mathbf{v} \in \mathbb{C}^4$ is*

$$\mathrm{circ}(\mathbf{v}) \;=\; \begin{pmatrix} v_0 & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_3 & v_2 \\ v_2 & v_1 & v_0 & v_3 \\ v_3 & v_2 & v_1 & v_0 \end{pmatrix}.$$

Note that this matrix is also Toeplitz due to the fact that it's constant along its diagonals (recall Exercise 3.7.1).

Exercise 3.7.4. *Show that every circulant matrix is also Toeplitz.*

Exercise 3.7.5. *Let $A \in \mathbb{C}^{2n \times 2n}$ be circulant. Show that $A_{j,k} = A_{j+n,k+n}$ for all $j,k \in [n]$. More generally, show that $A_{j,k} = A_{(j \pm n) \bmod 2n, (k \pm n) \bmod 2n}$ holds for all $j,k \in [2n]$.*

Not only is every circulant matrix a Toeplitz matrix, but any square Toeplitz matrix can be embedded into a larger circulant matrix. Hence, e.g., any algorithm which efficiently multiplies circulant matrices against vectors can also be used to efficiently multiply square Toeplitz matrices against vectors.

Let $\mathbf{a} \in \mathbb{C}^{2n-1}$ and consider the square Toeplitz matrix generated by $\mathbf{a}$, $\mathrm{Toep}_n(\mathbf{a}) \in \mathbb{C}^{n \times n}$, with entries given by

$$\left(\mathrm{Toep}_n(\mathbf{a})\right)_{j,k} = a_{(n-1)-(j-k)}. \tag{3.25}$$

Now let $\mathbf{c} \in \mathbb{C}^{2n}$ be defined by $\mathbf{c}^T = (a_{n-1}, a_{n-2}, \dots, a_0, 0, a_{2n-2}, \dots, a_n)$ so that

$$c_\ell = \begin{cases} a_{n-1-\ell} & 0 \le \ell \le n-1 \\ 0 & \ell = n \\ a_{3n-1-\ell} & n+1 \le \ell \le 2n-1 \end{cases} \tag{3.26}$$

for all $\ell \in [2n]$. Then, the circulant matrix $\mathrm{circ}(\mathbf{c}) \in \mathbb{C}^{2n \times 2n}$ will always take the block form

$$\mathrm{circ}(\mathbf{c}) = \begin{pmatrix} \mathrm{Toep}_n(\mathbf{a}) & A \\ A & \mathrm{Toep}_n(\mathbf{a}) \end{pmatrix} \in \mathbb{C}^{2n \times 2n}, \tag{3.27}$$

where $A \in \mathbb{C}^{n \times n}$.

Example 3.7.6. *Let $\mathbf{a} \in \mathbb{C}^3$. The 2×2 Toeplitz matrix generated by $\mathbf{a}$ is*

$$\mathrm{Toep}_2(\mathbf{a}) = \begin{pmatrix} a_1 & a_2 \\ a_0 & a_1 \end{pmatrix}.$$

If we form the vector $\mathbf{c} \in \mathbb{C}^4$ defined by $\mathbf{c}^T = (a_1, a_0, 0, a_2)$ then

$$\mathrm{circ}(\mathbf{c}) = \begin{pmatrix} a_1 & a_2 & 0 & a_0 \\ a_0 & a_1 & a_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ a_2 & 0 & a_0 & a_1 \end{pmatrix} = \begin{pmatrix} \mathrm{Toep}_2(\mathbf{a}) & A \\ A & \mathrm{Toep}_2(\mathbf{a}) \end{pmatrix} \in \mathbb{C}^{4 \times 4}.$$

The following lemma guarantees the upper-left $n \times n$ block of $\mathrm{circ}(\mathbf{c}) \in \mathbb{C}^{2n \times 2n}$ is indeed always $\mathrm{Toep}_n(\mathbf{a}) \in \mathbb{C}^{n \times n}$ as claimed above in (3.27).

Lemma 3.7.7. *Let $\mathbf{a} \in \mathbb{C}^{2n-1}$. Build $\mathbf{c} \in \mathbb{C}^{2n}$ from $\mathbf{a}$ entry-wise via (3.26). Then, $(\mathrm{circ}(\mathbf{c}))_{j,k} = (\mathrm{Toep}_n(\mathbf{a}))_{j,k}$ for all $j,k \in [n]$.*

Proof. We can see that $-(n-1) \leq j-k \leq (n-1)$ since $j, k \in [n]$. Furthermore,

$$(j-k) \bmod 2n = \begin{cases} j-k & 0 \leq j-k \leq n-1 \\ 2n+(j-k) & -(n-1) \leq j-k < 0 \end{cases}.$$

Hence, if $0 \leq j-k \leq n-1$ we have that

$$(\mathrm{circ}(\mathbf{c}))_{j,k} \ = \ c_{(j-k) \bmod n} \ = \ c_{j-k} \ = \ a_{n-1-(j-k)},$$

and if $-(n-1) \leq j-k < 0$ we have that

$$(\mathrm{circ}(\mathbf{c}))_{j,k} \ = \ c_{(j-k) \bmod n} \ = \ c_{2n+(j-k)} \ = \ a_{3n-1-(2n+(j-k))} \ = \ a_{n-1-(j-k)}.$$

Thus, $(\mathrm{circ}(\mathbf{c}))_{j,k} \ = \ (\mathrm{Toep}_N(\mathbf{a}))_{j,k} \ = \ a_{(n-1)-(j-k)}$ for all $j, k \in [n]$ by (3.25). □

Exercise 3.7.6. *Use Lemma 3.7.7 together with Exercise 3.7.5 to show that* (3.27) *holds for all $n \in \mathbb{N}$.*

Computationally, observe that via (3.27) we have for any $\mathbf{v} \in \mathbb{C}^n$ that

$$\mathrm{circ}(\mathbf{c}) \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \ = \ \begin{pmatrix} \mathrm{Toep}_n(\mathbf{a}) & A \\ A & \mathrm{Toep}_n(\mathbf{a}) \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \ = \ \begin{pmatrix} \mathrm{Toep}_n(\mathbf{a})\mathbf{v} \\ A\mathbf{v} \end{pmatrix}. \tag{3.28}$$

Thus, we can always recover $\mathrm{Toep}_n(\mathbf{a})\mathbf{v}$ from $\mathrm{circ}(\mathbf{c}) \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$ by taking its first n entries. Hence, as previously mentioned, any algorithm which efficiently multiplies circulant matrices against arbitrary vectors can also be used to efficiently multiply square Toeplitz matrices against arbitrary vectors. We will use this fact to our advantage later in Section 3.8.

3.7.2 Discrete Fourier Transforms and Circular Convolutions

In this section we will discuss a particular orthonormal basis of $\mathbb{C}^n$, known as the discrete Fourier basis, which is important for a large number of computational reasons involving convolutions. As we shall see, its many remarkable properties are in fact due to the periodic nature of the unit magnitude complex numbers $\{\mathbb{e}^{\mathbb{i}\theta} \mid \theta \in [0, 2\pi]\} \subset \mathbb{C}$. In particular, given $n \in \mathbb{N}$ the unit magnitude n^{th} root of unity

$$f_n := \mathbb{e}^{\frac{-2\pi\mathbb{i}}{n}} \in \mathbb{C}$$

will be the atomic building block of the basis, and its properties are therefore crucial.

Exercise 3.7.7. *Show that* $(f_n)^{kn} \ = \ f_n^{kn} \ = \ 1$ *for all $k \in \mathbb{Z}$.*

Exercise 3.7.8. *Show that* $(f_n)^k \ = \ f_n^k \ \neq \ 1$ *for all nonzero $k \in [n]$.*

Exercise 3.7.9. *Show that* $(f_n)^{\omega j} = f_n^{\omega j} = f_n^{(\omega \bmod n)(j \bmod n)} = f_n^{(\omega j \bmod n)}$ *for all* $j, \omega \in \mathbb{Z}$.[3]

Exercise 3.7.10. *Suppose that* $p, n \in \mathbb{N}$ *are such that* $\frac{n}{p} \in \mathbb{N}$ *(so that* p *divides* n*). Show that* $(f_n)^{pj} = f_n^{pj} = f_{\frac{n}{p}}^{(j \bmod \frac{n}{p})}$ *holds for all* $j \in \mathbb{Z}$.

Let $F \in \mathbb{C}^{n \times n}$ be the $n \times n$ matrix whose entries are given by

$$F_{\omega,j} := \frac{f_n^{\omega \cdot j}}{\sqrt{n}}$$

for all $\omega, j \in [n]$. The matrix F is called the **Discrete Fourier Transform (DFT) matrix of size** n. Importantly, the columns of F^* form an orthonormal basis of $\mathbb{C}^n$ (i.e., one can show that F is a unitary matrix – see Exercise 3.7.11). This basis is called the **discrete Fourier basis of** $\mathbb{C}^n$.

Example 3.7.8. *Recall that* $\mathbf{1} \in \mathbb{C}^n$ *denotes the vector of all ones. We have that*

$$F\mathbf{1} = \frac{1}{\sqrt{n}} \begin{pmatrix} \sum_{j=0}^{n-1} f_n^{0 \cdot j} \\ \sum_{j=0}^{n-1} f_n^{1 \cdot j} \\ \vdots \\ \sum_{j=0}^{n-1} f_n^{(n-1) \cdot j} \end{pmatrix}.$$

Considering the k^{th} *entry of* $F\mathbf{1} \in \mathbb{C}^n$ *for all* $k \neq 0$ *we can see that*

$$(F\mathbf{1})_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} f_n^{k \cdot j} = \frac{1}{\sqrt{n}} \left(\frac{1 - f_n^{kn}}{1 - f_n^k} \right) = \frac{1}{\sqrt{n}} \left(\frac{1 - 1}{1 - f_n^k} \right) = 0$$

by Exercises 3.7.7 and 3.7.8. On the other hand, for $k = 0$ *we have that*

$$(F\mathbf{1})_0 = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} f_n^{0 \cdot j} = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} 1 = \frac{n}{\sqrt{n}} = \sqrt{n}.$$

Hence, $F\mathbf{1} = \sqrt{n}\, \mathbf{e}_0$.

Exercise 3.7.11. *Prove that the DFT matrix,* F*, is unitary. (HINT: Recall Theorem 2.6.14.)*

Exercise 3.7.12. *Prove that* $\|F\mathbf{v}\|_2^2 = \|\mathbf{v}\|_2^2$ *holds for all* $\mathbf{v} \in \mathbb{C}^n$. *This equality is sometimes referred to as "Parseval's identity" in the context of the discrete Fourier basis.*

[3] Let $j \in \mathbb{Z}$. Recall that "j mod n" denotes the unique integer $r \in [n]$ satisfying $j = r + k \cdot n$ for some $k \in \mathbb{Z}$.

The **Discrete Fourier Transform (DFT)** of a vector $\mathbf{v} \in \mathbb{C}^n$ is simply

$$\widehat{\mathbf{v}} := F\mathbf{v} \tag{3.29}$$

with entries given by $\widehat{v}_\omega = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} v_j f_n^{\omega \cdot j}$ for all $\omega \in [n] = \{0, \dots, n-1\} \subset \mathbb{N}$. Similarly, the **Inverse Discrete Fourier Transform (IDFT)** of a vector $\mathbf{v} \in \mathbb{C}^n$ is

$$\widehat{\mathbf{v}}^{\text{-}1} := F^{-1}\mathbf{v} = F^*\mathbf{v}.$$

As we shall see, the DFT walks hand in hand with our next definition.

Exercise 3.7.13. *Suppose $p, n \in \mathbb{N}$ are such that $n/p \in \mathbb{N}$ (i.e., p divides n). Given $\mathbf{u} \in \mathbb{C}^p$, let $\mathbf{v} \in \mathbb{C}^n$ be a longer vector with entries given by*

$$v_j = \begin{cases} u_{pj/n} & \textit{if } j \equiv 0 \bmod (n/p) \\ 0 & \textit{else} \end{cases},$$

and let $\mathbf{w} \in \mathbb{C}^n$ be another longer vector with entries given by $w_j = u_{j\text{mod }p}$. Compute the n-length DFTs $\widehat{\mathbf{v}}, \widehat{\mathbf{w}} \in \mathbb{C}^n$ in terms of the p-length DFT of $\mathbf{u}$.

Exercise 3.7.14. *Let $a, b, c \in [n]$ be such that a is invertible modulo n.[4] Furthermore, suppose that $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$ satisfy*

$$v_j = \mathbb{e}^{\frac{2\pi \mathbb{i} cj}{n}} u_{aj+b \bmod n} \ = \ f_n^{-cj} u_{aj+b \bmod n}$$

for all $j \in [n]$. Write $\widehat{v}_\omega$ in terms of one or more entries of $\widehat{\mathbf{u}}$ for a given $\omega \in [n]$. How does a affect the entries of $\widehat{\mathbf{v}}$ when $c = b = 0$? How does b affect the entries of $\widehat{\mathbf{v}}$ when $a = 1$ and $c = 0$? How does c affect the entries of $\widehat{\mathbf{v}}$ when $a = 1$ and $b = 0$?

The **discrete (circular) convolution** of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$, denoted by $\mathbf{u} \star \mathbf{v} \in \mathbb{C}^n$, is defined entrywise via

$$(\mathbf{u} \star \mathbf{v})_k := \sum_{j=0}^{n-1} u_j \cdot v_{(k-j) \bmod n} \ = \ \sum_{j=0}^{n-1} u_{j \bmod n} \cdot v_{(k-j) \bmod n}$$

for all $k \in [n]$. Note that, in fact, $\mathbf{u} \star \mathbf{v} = \operatorname{circ}(\mathbf{v})\mathbf{u}$ for all $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$.

Example 3.7.9. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{C}^4$. Then,*

$$\mathbf{u} \star \mathbf{v} \ = \ \begin{pmatrix} \sum_{j=0}^3 u_j \ v_{-j \bmod n} \\ \sum_{j=0}^3 u_j \ v_{1-j \bmod n} \\ \sum_{j=0}^3 u_j \ v_{2-j \bmod n} \\ \sum_{j=0}^3 u_j \ v_{3-j \bmod n} \end{pmatrix} \ = \ \begin{pmatrix} v_0 & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_3 & v_2 \\ v_2 & v_1 & v_0 & v_3 \\ v_3 & v_2 & v_1 & v_0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} \ = \ \operatorname{circ}(\mathbf{v})\mathbf{u}.$$

[4] A value $a \in [n]$ is invertible modulo n if there exists an $h \in [n]$ such that $a\ h \equiv 1 \bmod n$. Any $a \in [n]$ that is relatively prime to n will be invertible modulo n by the Fermat-Euler Theorem (see, e.g., [36, Theorem 2.8]).

The discrete convolution has the following useful relationship with the discrete Fourier transform.

Theorem 3.7.10. *Let* $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$. *Then*

$$(\widehat{\mathbf{u} \star \mathbf{v}})_\omega = \sqrt{n}\; \widehat{u}_\omega \widehat{v}_\omega \tag{3.30}$$

holds for all $\omega \in [n]$.

Proof: To obtain (3.30) we compute

$$(\widehat{\mathbf{u} \star \mathbf{v}})_\omega \;=\; \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (\mathbf{u} \star \mathbf{v})_k \, f_n^{\omega \cdot k} \;=\; \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \left(\sum_{j=0}^{n-1} u_j \cdot v_{(k-j) \bmod n} \right) f_n^{\omega \cdot k}.$$

Exchanging the final double sum we obtain that

$$(\widehat{\mathbf{u} \star \mathbf{v}})_\omega \;=\; \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} u_j \, f_n^{\omega \cdot j} \left(\sum_{k=0}^{n-1} v_{(k-j) \bmod n} \, f_n^{\omega \cdot (k-j)} \right) \;=\; \sqrt{n}\; \widehat{u}_\omega \widehat{v}_\omega.$$

Here we have used the fact that $f_n^{\ell \cdot n} = 1$ for all $\ell \in \mathbb{Z}$ so that $f_n^{\omega \cdot (k-j)} = f_n^{\omega \cdot ((k-j) \bmod n)}$ always holds (see, e.g, Exercise 3.7.9). □

Exercise 3.7.15. *Show that* $\mathrm{circ}(\mathbf{u})\mathbf{v} \;=\; \mathbf{v} \star \mathbf{u} \;=\; \mathbf{u} \star \mathbf{v} \;=\; \mathrm{circ}(\mathbf{v})\mathbf{u}$ *holds for all* $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$.

Theorem 3.7.10 tells us that the DFT of the convolution of two vectors is, up to rescaling by $\sqrt{n}$, equal to the entrywise product of the DFTs of the two vectors. Using this relationship we can compute the discrete convolution of $\mathbf{u}$ and $\mathbf{v}$ using their DFTs. Let $\mathbf{u} \odot \mathbf{v} \in \mathbb{C}^n$ denote the entrywise (or Hadamard) product of the two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$. That is, let

$$(\mathbf{u} \odot \mathbf{v})_j := u_j v_j$$

for all $j \in [n]$. Theorem 3.7.10 now directly implies that

$$\mathbf{u} \star \mathbf{v} \;=\; \sqrt{n}\; \widehat{\widehat{\mathbf{u}} \odot \widehat{\mathbf{v}}}^{\,-1} \;=\; \sqrt{n}\; F^* \left(F\mathbf{u} \odot F\mathbf{v} \right). \tag{3.31}$$

Note that the last expression of (3.31) could be computed quickly if we could find a way to quickly calculate both $F\mathbf{u}$ and $F^*\mathbf{u}$ for any given $\mathbf{u}$. This is in fact possible as we shall see in Section 3.8.

The following additional fact relating IDFT matrices to circulant matrices is closely related to Theorem 3.7.10: Every column of the IDFT matrix F^* is an eigenvector of every circulant matrix. As a result, the $n \times n$ IDFT matrix F^* simultaneously diagonalizes this entire class of $n \times n$ matrices.

Theorem 3.7.11. *Let $\mathbf{v} \in \mathbb{C}^n$. Every column of $F^* \in \mathbb{C}^{n \times n}$ is an eigenvector of* $\operatorname{circ}(\mathbf{v})$*.*

Proof. Let $\mathbf{u} = F^* \mathbf{e}_j \in \mathbb{C}^n$ be the j^{th} column of F^*. By (3.31),

$$\begin{aligned}\operatorname{circ}(\mathbf{v})\mathbf{u} \;&=\; \mathbf{u} \star \mathbf{v} \;=\; \sqrt{n}\; F^*(F\mathbf{u} \odot F\mathbf{v}) \;=\; \sqrt{n}\; F^*((FF^*\mathbf{e}_j) \odot F\mathbf{v}) \\ &=\; \sqrt{n}\; F^*(\mathbf{e}_j \odot \widehat{\mathbf{v}}) \;=\; \sqrt{n}\; F^*(\widehat{v}_j \mathbf{e}_j) \;=\; \sqrt{n}\; \widehat{v}_j \mathbf{u}\end{aligned}$$

Thus, the j^{th} column of F^* is an eigenvector of $\operatorname{circ}(\mathbf{v})$ with eigenvalue $\sqrt{n}\, \widehat{v}_j$. □

Exercise 3.7.16. *Let $\mathbf{v} \in \mathbb{C}^n$. Show that* $\operatorname{circ}(\mathbf{v}) \in \mathbb{C}^{n \times n}$ *is invertible if and only if $\widehat{v}_\omega \neq 0$ for all $\omega \in [n]$.*

Exercise 3.7.17. *Order the Fourier coefficients of $\mathbf{v} \in \mathbb{C}^n$ by magnitude so that*

$$|\widehat{v}_{\omega_0}| \geq |\widehat{v}_{\omega_1}| \geq \cdots \geq |\widehat{v}_{\omega_{n-1}}|.$$

Prove that the j^{th} singular value of $\operatorname{circ}(\mathbf{v}) \in \mathbb{C}^{n \times n}$ *satisfies $\sigma_j\left(\operatorname{circ}(\mathbf{v})\right) = \sqrt{n}\; |\widehat{v}_{\omega_j}|$.*

One important consequence of the proof above is that the DFT gives us an easy way to compute the eigenvalues of all circulant matrices. Of even more consequence, though, is that (3.31) can also be used in many other applications where convolutions naturally appear. We have already seen, e.g., that the Toeplitz weight matrices of convolutional layers of neurons can be embedded into circulant matrices (recall definition 3.7.3 and (3.27)). Hence, (3.31) can potentially help evaluate convolutional layers of neurons more quickly via (3.28). In addition, convolutions also appear in numerous other important applications, two of which we will briefly discuss next.

Example 3.7.12 (Deblurring)**.** *Consider the following "deblurring" problem: given $\mathbf{u} \star \mathbf{v} \in \mathbb{C}^n$ (the blurry signal) and knowledge of the blur kernel $\mathbf{v} \in \mathbb{C}^n$ (e.g., a Gaussian blur kernel), recover the unblurred signal $\mathbf{u} \in \mathbb{C}^n$. Such problems are common in imaging applications where a blurred image can indeed be thought of as a crisp/unblurred imaged convolved with a blur kernel. The question then becomes how one can try to "undo the blur" in order to get $\mathbf{u} \in \mathbb{C}^n$ back from its blurry version $\mathbf{u} \star \mathbf{v} \in \mathbb{C}^n$.*

Somewhat amazingly, this is easy to do efficiently <u>if</u> we have both the blurry signal $\mathbf{u} \star \mathbf{v} \in \mathbb{C}^n$ <u>and</u> knowledge of how the original image was likely blurred (i.e., we also know $\mathbf{v} \in \mathbb{C}^n$). In that case one can compute

$$\widehat{u}_\omega \;=\; \frac{\widehat{\mathbf{u} \star \mathbf{v}}_\omega}{\widehat{v}_\omega}$$

for all $\omega \in [n]$, and then set $\mathbf{u} = F^* \widehat{\mathbf{u}}$. This of course assumes that the Fourier coefficients $\widehat{v}_\omega \neq 0$ for all $\omega \in [n]$. If there are zero Fourier coefficients, then one can instead note that we are equivalently simply trying to solve the linear system $\operatorname{circ}(\mathbf{v})\mathbf{u} = \mathbf{u} \star \mathbf{v}$ *for $\mathbf{u} \in \mathbb{C}^n$. In such a case we can instead <u>always</u> find an approximate solution by returning, e.g., the least-squares estimate $\tilde{\mathbf{u}} \;=\; \operatorname{circ}(\mathbf{v})^\dagger(\mathbf{u} \star \mathbf{v}) \;=\; P_{\mathcal{C}(\operatorname{circ}(\mathbf{v})^*)}\mathbf{u}$ (recall Section 3.2). Furthermore, an SVD of* $\operatorname{circ}(\mathbf{v})$*, and therefore* $\operatorname{circ}(\mathbf{v})^\dagger$*, can be constructed efficiently using Theorem 3.7.11.*

Example 3.7.13 (Polynomial Multiplication)**.** *Convolutions also appear naturally as part of polynomial multiplication. Let* $q(x) = \sum_{j=0}^{n-1} q_j x^j$ *and* $r(x) = \sum_{j=0}^{n-1} r_j x^j$ *be two polynomials. Then* $t(x) = q(x) \cdot r(x)$ *is a polynomial of degree* $\leq 2n-2$ *that can be expressed as* $t(x) = \sum_{j=0}^{2n-2} t_j x^j$*. Writing the coefficients of* q *and* r *as vectors* $\mathbf{q}, \mathbf{r} \in \mathbb{C}^n$*, respectively, and the coefficients of* t *as a vector* $\mathbf{t} \in \mathbb{C}^{2n-1}$*, we have that*

$$\mathbf{t} = \begin{pmatrix} \mathbf{q} \\ \mathbf{0} \end{pmatrix} \star \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix}.$$

For example, when $n = 3$ *we have that*

$$\begin{aligned} t(x) &= (q_2x^2 + q_1x + q_0)(r_2x^2 + r_1x + r_0) \\ &= \underbrace{q_2r_2}_{t_4} x^4 + \underbrace{(q_2r_1 + q_1r_2)}_{t_3} x^3 + \underbrace{(q_2r_0 + q_1r_1 + q_0r_2)}_{t_2} x^2 + \underbrace{(q_1r_0 + q_0r_1)}_{t_1} x + \underbrace{q_0r_0}_{t_0}. \end{aligned}$$

In vector form this corresponds to

$$\begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} q_0 & 0 & 0 & q_2 & q_1 \\ q_1 & q_0 & 0 & 0 & q_2 \\ q_2 & q_1 & q_0 & 0 & 0 \\ 0 & q_2 & q_1 & q_0 & 0 \\ 0 & 0 & q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix} = \text{circ}\left(\begin{pmatrix} \mathbf{q} \\ \mathbf{0} \end{pmatrix}\right) \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ 0 \\ 0 \end{pmatrix} \star \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix}.$$

Hence, if we can compute (I)DFTs quickly then we can also multiply polynomials quickly via (3.31)*.*

Exercise 3.7.18. *Consider the "finite difference" matrix* $D_2 \in \mathbb{R}^{n \times n}$ *whose entries are given by*

$$(D_2)_{i,j} = \begin{cases} -2 & \text{if } i = j \\ 1 & \text{if } (i-j) \equiv 1 \text{ mod } n \\ 1 & \text{if } (i-j) \equiv n-1 \text{ mod } n \\ 0 & \text{otherwise} \end{cases}. \tag{3.32}$$

This is an example of a circulant matrix. Show that $FD_2 = EF$*, where* $E \in \mathbb{R}^{n \times n}$ *is a diagonal matrix with entries given by*

$$(E)_{i,j} = \begin{cases} 2\cos(2\pi j/n) - 2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \tag{3.33}$$

Exercise 3.7.19. *Let* $D_{2r} \in \mathbb{R}^{n \times n}$ *be defined by* $D_{2r} := D_2^r$*. Use the previous exercise to show that* $FD_{2r} = E^r F$ *for all* $r \in \mathbb{Z}^+$*.*

As we will discuss in the next section, there is indeed a fast algorithm for computing both $F\mathbf{u}$ and $F^*\mathbf{u}$ for all $\mathbf{u} \in \mathbb{C}^n$. As a result, there are fast (i.e., computationally efficient) algorithms based on (3.31) for rapidly computing the convolutions involved in all of the applications mentioned in this section. Before explaining how any of these fast algorithms work, however, let's first briefly discuss what we actually mean when we say an algorithm is "fast".

3.7.3 Big-$\mathcal{O}$ Notation and the Basic Art of Runtime Analysis

Throughout this text we will approach runtime discussions/analysis by counting six general types of atomic computational operations which we will assume any reasonable computer can do in a constant amount of time. These six types of constant-cost operations are:

1. Assigning a complex value to/reading a complex value from a variable or vector entry (e.g., setting $x_j = y \in \mathbb{C}$).

2. Adding/subtracting two machine numbers (e.g., adding any two real or complex numbers to a fixed precision).

3. Multiplying/dividing two machine numbers (e.g., multiplying any two real or complex numbers to a fixed precision).

4. Comparing two machine numbers (e.g., deciding whether one real number is larger/smaller/ equal to another real number).

5. Evaluating basic logical expressions and conditional statements (e.g., deciding if "(boolean value A) AND/OR (boolean value B)" is True or False).

6. Evaluating simple functions $f : \mathbb{R} \rightarrow \mathbb{R}$ to a fixed precision. Herein, this class of "simple functions" includes (*i*) functions with rapidly convergent Maclaurin (i.e., 0-centered Taylor) series expansions such as the exponential, sine, and cosine functions, (*ii*) related complex-valued functions like $\mathbb{e}^{\mathbb{i}\theta} = \cos(\theta) + \mathbb{i}\sin(\theta)$, and (*iii*) other rapidly approximable functions such as $f(t) = t^{\alpha}$ for a given (e.g., non-integer) $\alpha \in \mathbb{R}$.

Looking at the "constant-cost" operations above the invested reader's eyebrows should be at least slightly raised. The sixth type of operation (evaluating simple functions) seems particularly fishy, doesn't it?[5] Even the second type of operation (i.e., simple addition) being "constant-cost" should inculcate suspicion in anyone who was expected to add 6 digit numbers to one another by hand in elementary school. I urge anyone who is not skeptical to grab a piece of of chalk and investigate the claim that adding two 300 digit numbers together is the same "constant-cost" operation as adding 9 to 8.[6] That said, let me urge you to allow the escape clause "to a fixed precision", as well as the related term "machine numbers", to save you from your skepticism, at least enough to believe that there is indeed some value to such simple types of operation counts.

[5] We urge the interested reader to consult, e.g,. [31, Chapters 1 and 3] to learn why this sixth type of constant-cost operation is indeed not *too* fishy after all....

[6] Really even adding two numbers should not be considered "constant-time" if you are doing serious computations involving, e.g., large number (and, therefore, extended precision) arithmetic. More precisely, the complexity of addition should depend on the the number of digits in each sum that you want to be able to correctly compute. Of course, this type of more complicated accounting then only gets more involved as you consider the other types of operations above.

Generally speaking, a digital computer can only guarantee the calculation of a fixed number of the leading digits of any real number one aims to compute/store. This is simply a fact of life. All of our algorithms here (or any others you see that are analyzed in a similar way) only guarantee you numerical answers up to some precision, or number of digits of accuracy – if that number of digits is not enough to be meaningful, then the algorithms are computing garbage. If, however, the answer you are after can be expressed accurately enough to satisfy you by its most significant, e.g., ~ 16 decimal digits, then the type of accounting we do here will be completely adequate for you. Even if you want many more digits of accuracy, though, a computer algorithm that needs to use only a few higher-precision operations will still be much faster to execute that one that uses many more higher-precision operations. As a result, even if our operation counts don't truly represent an algorithm's runtime complexity with 100% accuracy in all cases (they don't), they do at least correlate well enough to be informative.[7]

Example 3.7.14 (Matrix-vector Multiplication)**.** *As an illustrative example, let's consider the runtime complexity of computing the matrix-vector product $A\mathbf{x} \in \mathbb{C}^m$ for an arbitrary matrix $A \in \mathbb{C}^{m\times n}$ and vector $\mathbf{x} \in \mathbb{C}^n$. Noting that each entry of $\mathbf{y} = A\mathbf{x} \in \mathbb{C}^m$ is computed by*

$$y_j = (A\mathbf{x})_j = \sum_{k\in[n]} A_{j,k}x_k,$$

we can can see that calculating $y_j \in \mathbb{C}$ requires $4n$ operations (we must read the $2n$ $A_{j,k}/x_k$ values into memory and then perform n multiplications, $n-1$ additions, and finally one assignment of the correct value to y_j). Given that we must compute y_j for all $j \in [m]$ in order to calculate $\mathbf{y} = A\mathbf{x}$ we can now conclude that computing $\mathbf{y}$ will require at most $4nm$ constant-cost operations.[8]

In the example above the constant 4 we ended up with matters much less in general than the parameters m and n which will be significantly larger than 4 for big matrices A. As a result, it's standard practice to simplify operation counts by ignoring all such constants via big-$\mathcal{O}$ notation.

Definition 3.7.15 (Big-$\mathcal{O}$ Notation)**.** *Let $f, g : (0,1)^n \times (1,\infty)^m \to [0,\infty)$ be two functions of $n+m \geq 1$ variables for nonnegative $n, m \in \mathbb{Z}$. We say that f is $\mathcal{O}(g)$ if there exists a constant $C \in [1,\infty)$ and values $(\delta_0,\ldots,\delta_{n-1}) \times (y_0,\ldots,y_{m-1}) \in (0,1)^n \times (1,\infty)^m$ such that*

$$f(\epsilon_0,\ldots,\epsilon_{n-1},x_0,\ldots,x_{m-1}) \leq Cg(\epsilon_0,\ldots,\epsilon_{n-1},x_0,\ldots,x_{m-1})$$

whenever $\epsilon_j < \delta_j$ and $x_k > y_k$ hold for all $j \in [n]$ and $k \in [m]$.

[7] We urge the interested reader to consult, e.g, [31, Chapter 2] and [15, Chapters 2 and 3] to learn more about numerical precision, machine numbers, and algorithmic runtime analysis. To begin understanding how one might make complexity analysis more rigorous one can also consult, e.g., [37].

[8] Here we say that computing $\mathbf{y}$ will require *at most* $4nm$ constant cost operations because we have demonstrated a way to compute $\mathbf{y}$ using this number of operations. However, there might be better ways to do it that use fewer operations by, e.g., avoiding rereading x_k multiple times for every different y_j calculation.

We can now see from Example 3.7.14 that computing $A\mathbf{x}$ can always be done using $\mathcal{O}(mn)$ operations (check!). The following example will also help illustrate the proper use of big-$\mathcal{O}$ notation.

Example 3.7.16. *Consider* $f, g : (0,1)^2 \times (1,\infty)^2 \to \mathbb{R}^+$ *given by* $f(\epsilon, \delta, m, n) = n/\epsilon + m/\delta + 100n + 200$ *and* $g(\epsilon, \delta, m, n) = \frac{\max\{m,n\}}{\min\{\epsilon,\delta\}}$. *We will show that* f *is* $\mathcal{O}(g)$.

To start we note that $1 < \frac{1}{\min\{\epsilon,\delta\}}$ *holds for all* $\epsilon, \delta \in (0,1)$. *Similarly,* $1 < \max\{m,n\}$ *holds for all* $m, n \in (1,\infty)$. *Hence,*

$$\begin{aligned} f(\epsilon,\delta,m,n) &= n/\epsilon + m/\delta + 100n + 200 \ \leq \ \max\{m,n\}\left(\frac{1}{\epsilon} + \frac{1}{\delta} + 100\right) + 200 \\ &\leq \ \max\{m,n\}\left(\frac{2}{\min\{\epsilon,\delta\}} + 100\right) + 200 \ \leq \ \max\{m,n\}\left(\frac{102}{\min\{\epsilon,\delta\}}\right) + 200 \\ &\leq \ 302\left(\frac{\max\{m,n\}}{\min\{\epsilon,\delta\}}\right) \ = \ 302 \cdot g(\epsilon,\delta,m,n) \end{aligned}$$

holds for all $(\epsilon,\delta) \times (m,n) \in (0,1)^2 \times (1,\infty)^2$. *As a consequence, we can see that* f *is indeed* $\mathcal{O}(g)$ *in accordance with Definition 3.7.15.*

Exercise 3.7.20. *Let* $g : (0,1) \times [1,\infty)^2 \to [0,\infty)$ *be given by* $g(\epsilon, x, y) = \frac{x \log y}{\epsilon}$. *Which of these functions* $f : (0,1) \times [1,\infty)^2 \to [0,\infty)$ *are* $\mathcal{O}(g)$*? Explain your answers.*

(a) $f(\epsilon, x, y) = \frac{300x \log y}{\epsilon} + 50$

(b) $f(\epsilon, x, y) = 500x^{0.34} + 600/\epsilon + 10^6 \log y$

(c) $f(\epsilon, x, y) = \frac{0.2\ x}{\epsilon^2} + \log y$

(d) $f(\epsilon, x, y) = \frac{20\sqrt{x}\log(100+\log(y))}{\sqrt{\epsilon}}$

Exercise 3.7.21. *Let* $A \in \mathbb{C}^{m \times n}$ *and* $B \in \mathbb{C}^{n \times p}$. *Show that computing* $AB \in \mathbb{C}^{m \times p}$ *can be done using* $\mathcal{O}(mnp)$ *operations.*[9]

3.8 The Fast Fourier Transform (FFT)

As seen above, computing the DFT of a vector $\mathbf{v} \in \mathbb{C}^n$ requires the computation of $F\mathbf{v}$. Computing $F\mathbf{v}$ directly via a generic matrix-vector multiply as per Example 3.7.14 uses $\mathcal{O}(n^2)$ operations. In this section we will discuss the Fast Fourier Transform (FFT) algorithm which can compute the DFT of a vector using only $\mathcal{O}(n \log n)$ operations. Though this reduction in computational complexity might seem slight at first glance, this speedup

[9] There are in fact faster (though not terribly practical) matrix multiplication algorithms out there for arbitrary matrices! We direct the interested reader to, e.g., [15, Chapter 28],[45, 28, 48].

has had such far reaching impacts that the FFT has been lauded as one of the ten most important algorithmic developments of the twentieth century as a result [13].[10]

The FFT was first published and analyzed as a computer algorithm by Cooley and Tukey in 1965 [14], despite similar techniques being utilized much earlier (e.g., by Gauss and many others [23]). Cooley and Tukey's algorithm is particularly efficient for vector dimensions, n, whose prime factorizations contain only small prime factors. Later variants of the FFT [5, 39] allowed the FFT to also be utilized effectively for vector sizes whose prime factorizations contain larger primes. This section has primarily follows [14, 5, 39]. For more information on Fourier methods and algorithms we recommend that the interested reader consult the relevant chapters of [38], [31], [15], or [7]. For a fast FFT implementation we recommend FFTW [19]. In what follows we will outline the recursive construction of the FFT algorithm via sum splitting techniques.

Let $\mathbf{u} \in \mathbb{C}^n$, and suppose that its dimension, n, has the prime factorization

$$n = p_1 \cdot p_2 \cdots p_m, \text{ where } p_1 \leq p_2 \leq \cdots \leq p_m \text{ are } n\text{'s prime factors.}$$

Choose $\omega \in [n]$. Recalling the definition of the DFT we have that

$$\widehat{u}_\omega = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} u_j \, f_n^{\omega \cdot j}. \tag{3.34}$$

By splitting the sum (3.34) for $\widehat{u}_\omega$ into p_1 smaller subsums, one for each possible residue modulo p_1, we can see that

$$\widehat{u}_\omega = \frac{1}{\sqrt{n}} \sum_{k=0}^{p_1-1} f_n^{\omega \cdot k} \left(\sum_{j=0}^{\frac{n}{p_1}-1} u_{k+p_1 \cdot j} \, f_n^{\omega \cdot p_1 \cdot j} \right). \tag{3.35}$$

Let's now rewrite the internal sum of (3.35) in order to realize some progress.

Given $k \in [p_1]$, define $\mathbf{u}^{(k,p_1)} \in \mathbb{C}^{n/p_1}$ to be the vector whose entries are the entries of $\mathbf{u}$ having indexes that are congruent to k modulo p_1,

$$\left(\mathbf{u}^{(k,p_1)}\right)_j := u_{k+j \cdot p_1} \tag{3.36}$$

[10] The QR decomposition discussed in Section 2.4 also made the list of the top 10 most important algorithmic developments by the way!

for all $j \in [n/p_1]$.[11] Our equation (3.35) for $\widehat{u}_\omega$ now becomes

$$\widehat{u}_\omega = \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega \cdot k} \left(\frac{1}{\sqrt{n/p_1}} \sum_{j=0}^{\frac{n}{p_1}-1} \left(\mathbf{u}^{(k,p_1)} \right)_j f_n^{p_1 \cdot \omega \cdot j} \right) \tag{3.37}$$

$$= \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega \cdot k} \left(\frac{1}{\sqrt{n/p_1}} \sum_{j=0}^{\frac{n}{p_1}-1} \left(\mathbf{u}^{(k,p_1)} \right)_j f_{\frac{n}{p_1}}^{\left(\omega \bmod \frac{n}{p_1}\right) \cdot j} \right)$$

$$= \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega \cdot k} \left(\widehat{\mathbf{u}^{(k,p_1)}} \right)_{\omega \bmod \frac{n}{p_1}} . \tag{3.38}$$

For the sake of clarity we emphasize that the vector $\widehat{\mathbf{u}^{(k,p_1)}} \in \mathbb{C}^{n/p_1}$ in (3.38) is exactly $F\mathbf{u}^{(k,p_1)}$, where $F \in \mathbb{C}^{\frac{n}{p_1} \times \frac{n}{p_1}}$ is now the DFT matrix of size n/p_1. We strongly recommend that you verify the equality of (3.37) – (3.38) for yourself before reading further.

At this point it's useful to ask ourselves what we've managed to accomplish by reformulating (3.34) into (3.38). Mainly, we can now compute $\widehat{\mathbf{u}} \in \mathbb{C}^n$ with fewer operations than before by computing it in two steps. First, we compute $\widehat{\mathbf{u}^{(k,p_1)}} \in \mathbb{C}^{\frac{n}{p_1}}$ for all $k \in [p_1]$. Next, we use the vectors $\widehat{\mathbf{u}^{(0,p_1)}}, \dots, \widehat{\mathbf{u}^{(p_1-1,p_1)}}$ computed in the first step in order to compute each entry of $\widehat{\mathbf{u}}$ via (3.38). The first step can be accomplished with p_1 matrix-vector multiplications, each of which can be computed using $\mathcal{O}(n^2/p_1^2)$ operations (recall that $\widehat{\mathbf{u}^{(k,p_1)}} = F\mathbf{u}^{(k,p_1)}$, where F is the DFT matrix of size n/p_1). Hence, the first step can be completed using $\mathcal{O}(n^2/p_1)$ total operations. Step two only requires $\mathcal{O}(p_1 n)$ total operations in order to finish calculating $\widehat{\mathbf{u}}$, $\mathcal{O}(p_1)$-operations for each entry $\widehat{u}_\omega$. Putting it all together, we can see that (3.38) allows us compute $\widehat{\mathbf{u}} \in \mathbb{C}^n$ using a grand total of $\mathcal{O}(p_1 n + n^2/p_1)$ operations, as opposed to computing it directly via (3.29) using $\sim n^2$ operations.

Although the computational gain obtained from (3.38) is modest when $p_1 \ll n$, it is important to note that the sum-splitting technique used to obtain it can now be employed again in order to compute each $\widehat{\mathbf{u}^{(k,p_1)}}$, $k \in [0, p_1)$, more quickly. That is, we may split up the sum for $(\widehat{\mathbf{u}^{(k,p_1)}})_\omega$ into p_2 additional sums, etc.. Repeatedly sum-splitting in this fashion leads to the recursive **Fast Fourier Transform (FFT)** shown in Algorithm 6. Analogous sum-splitting leads to the **Inverse Fast Fourier Transform (IFFT)** which can be obtained from Algorithm 6 by replacing line 10's $f_n^{k\omega}$ by $f_n^{-k\omega}$ and replacing each $\widehat{\mathbf{u}}$ by a $\widehat{\mathbf{u}}^{-1}$.

We are now ready to analyze the computational complexity of the FFT. Let T_n be the total number of operations used by Algorithm 6 to compute $\widehat{\mathbf{u}} \in \mathbb{C}^n$. In order to determine an equation for T_n we note that lines 6 – 8 of Algorithm 6 require $p_1 \cdot T_{\frac{n}{p_1}}$ operations while

[11] Note that we used an integer divisor of n, i.e. p_1, exactly to ensure that $\frac{n}{p_1} \in \mathbb{N}$.

Algorithm 6 A Recursive Fast Fourier Transform (FFT) Implementation

```
1: Input: $\mathbf{u} \in \mathbb{C}^n$, Dimension $n$, and $n$'s prime factorization $p_1 \leq \cdots \leq p_m$
2: Output: $\widehat{\mathbf{u}} \in \mathbb{C}^n$
3: if $n = 1$ then
4:     Return $\mathbf{u}$
5: end if
6: for $k$ from 0 to $p_1 - 1$ do
7:     $\widehat{\mathbf{u}^{(k,p_1)}} \leftarrow \text{FFT}\left(\mathbf{u}^{(k,p_1)}, \frac{n}{p_1}, \; p_2 \leq p_3 \leq \cdots \leq p_m\right)$
8: end for
9: for $\omega$ from 0 to $n - 1$ do
10:     $\widehat{u}_\omega \leftarrow \frac{1}{\sqrt{p_1}} \left( \sum_{k=0}^{p_1 - 1} f_n^{k\omega} \left(\widehat{\mathbf{u}^{(k,p_1)}}\right)_{\omega \bmod \frac{n}{p_1}} \right)$
11: end for
12: Return $\widehat{\mathbf{u}}$
```

lines 9 – 11 use $\mathcal{O}(p_1 n)$ operations. Therefore we have

$$T_n = \mathcal{O}(p_1 n) + p_1 \cdot T_{\frac{n}{p_1}}.$$

However, Algorithm 6 is recursively invoked again to compute $\widehat{\mathbf{u}^{(0,p_1)}}, \dots, \widehat{\mathbf{u}^{(p_1-1,p_1)}}$ by sum-splitting in line 7. Taking this into account we can see that

$$T_{\frac{n}{p_1}} = \mathcal{O}\left(p_2 \frac{n}{p_1}\right) + p_2 \cdot T_{\frac{n}{p_1 p_2}}.$$

We now have that

$$T_n = \mathcal{O}(p_1 n) + p_1 \cdot \left(\mathcal{O}\left(\frac{p_2 n}{p_1}\right) + p_2 \cdot T_{\frac{n}{p_1 p_2}} \right) = \mathcal{O}\left(n(p_1 + p_2)\right) + p_1 p_2 \cdot T_{\frac{n}{p_1 p_2}}.$$

Repeating this recursive sum-splitting $j \leq m$ times shows us that

$$T_n = \mathcal{O}\left(n \cdot \sum_{\ell=1}^{j} p_\ell\right) + \prod_{\ell=1}^{j} p_\ell \cdot T_{\frac{n}{p_1 \cdots p_j}}.$$

Using that $T_1 = \mathcal{O}(1)$ (see Algorithm 6's lines 3 – 5) we have

$$T_n = \mathcal{O}\left(n \cdot \sum_{\ell=1}^{m} p_\ell\right) + \mathcal{O}(n) = \mathcal{O}(m \cdot p_m \cdot n). \tag{3.39}$$

Note that $m \leq \log_2 n$ while p_m is n's largest prime factor. We have proven the following theorem in the course of the prior discussion.

Theorem 3.8.1. *Let* $\mathbf{u} \in \mathbb{C}^n$ *and suppose that* $n \in \mathbb{N}$ *has the prime factorization* $n = p_1 \cdots p_m$*, where* $p_1 \leq p_2 \leq \cdots \leq p_m$ *are the prime factors of* n *ordered from smallest to largest. Then, we may compute* $\widehat{\mathbf{u}} = F\mathbf{u}$ *using* $\mathcal{O}\left(n \cdot \sum_{\ell=1}^{m} p_\ell\right)$ *operations.*

Theorem 3.8.1 tells us that the FFT can significantly speed up computation of the DFT. For example, if n is a power of 2 we'll have $m = \log_2 n$ and $p_m = 2$ leaving Algorithm 6 with an $\mathcal{O}(n \ln n)$ operation count. This is a clear improvement over the $\sim n^2$ operations required to in order to compute (3.29) directly. However, if n has large prime factors the improvement is less impressive. In the worst case, when n is itself a prime number, we have $m = 1$ and $p_1 = n$. This leaves Algorithm 6 with a $\mathcal{O}(n^2)$ runtime which, in practice, is even slower than the direct method (3.29).

The inability of Algorithm 6 to efficiently handle vectors with sizes containing large prime factors isn't a setback when one may dictate, with little or no repercussions, the dimension of the vectors they work with. A popular choice is to simply force n to be a power of 2. However, sometimes one simply needs to compute the DFT of a vector whose size contains (or may contain) large prime factors. In the next subsection we discuss how to do this efficiently.

Exercise 3.8.1 (Computational Exercise)**.** *Implement both the FFT and the IFFT for vectors of size* 2^n*,* $n \in \mathbb{N}$*. Produce a plot showing that each is indeed faster than the corresponding naive method for directly computing the (I)DFT of an arbitrary vector.*

3.8.1 The FFT for Vectors of Arbitrary Size

As discussed in the previous subsection, Algorithm 6 may not be a very efficient means of computing $\widehat{\mathbf{u}} \in \mathbb{C}^n$ when n contains large prime factors. One way of addressing this issue is to rewrite $\widehat{\mathbf{u}}$ as a discrete convolution of two vectors of a slightly larger dimension, $\tilde{n} > n$, that does contain only small prime factors. This discrete convolution can then be computed quickly using Algorithm 6 which will be efficient for vectors of dimension $\tilde{n}$.

Let $\omega \in [n]$. We may rewrite $\widehat{u}_\omega$ as

$$\widehat{u}_\omega \;=\; f_n^{\frac{\omega^2}{2}} f_n^{-\frac{\omega^2}{2}} \cdot \widehat{u}_\omega \;=\; \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot \sum_{j=0}^{n-1} u_j \, f_n^{\omega \cdot j - \frac{\omega^2}{2}} \;=\; \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot \sum_{j=0}^{n-1} u_j \, f_n^{\frac{j^2}{2}} \, f_n^{\frac{-(\omega-j)^2}{2}} \tag{3.40}$$

Note that the last sum in (3.40) resembles a convolution. In order to make the resemblance more concrete we will define two new vectors.

Let $\tilde{n} = 2^{\lceil \log_2 n \rceil + 1} \geq 2n$ and define $\tilde{\mathbf{u}} \in \mathbb{C}^{\tilde{n}}$ by

$$\tilde{u}_j := \begin{cases} u_j \cdot f_n^{\frac{j^2}{2}} & \text{if } 0 \leq j < n \\ 0 & \text{if } n \leq j < \tilde{n} \end{cases},$$

and $\mathbf{v} \in \mathbb{C}^{\tilde{n}}$ by

$$v_h := \begin{cases} f_n^{\frac{-h^2}{2}} & \text{if } 0 \leq h < n \\ 0 & \text{if } n \leq h \leq \tilde{n} - n \\ f_n^{\frac{-(h-\tilde{n})^2}{2}} & \text{if } \tilde{n} - n < h < \tilde{n} \end{cases} .$$

Computing a weighted convolution of $\tilde{\mathbf{u}}, \mathbf{v} \in \mathbb{C}^{\tilde{n}}$ we can see that

$$\begin{aligned} \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot (\tilde{\mathbf{u}} \star \mathbf{v})_\omega &= \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \sum_{j=0}^{\tilde{n}-1} \tilde{u}_j \cdot v_{(\omega - j) \bmod \tilde{n}} \\ &= \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \left(\sum_{j=0}^{\omega} \tilde{u}_j \cdot v_{\omega - j} + \sum_{j=\omega+1}^{n-1} \tilde{u}_j \cdot v_{(\omega-j)+\tilde{n}} \right) \\ &= \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \sum_{j=0}^{n-1} u_j \cdot f_n^{\frac{j^2}{2}} f_n^{\frac{-(\omega-j)^2}{2}} . \end{aligned}$$

Comparing to (3.40) now reveals that

$$\widehat{u}_\omega = \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot (\tilde{\mathbf{u}} \star \mathbf{v})_\omega \quad \forall \omega \in [n]. \tag{3.41}$$

Note that the convolution in (3.41) can be computed efficiently by the FFT and IFFT using (3.31) since $\tilde{n}$ is a power of two. Furthermore, $\tilde{n} \leq 4n$ by definition. Hence, we have established the following theorem.

Theorem 3.8.2. *Let $\boldsymbol{u} \in \mathbb{C}^n$. Then, both $\widehat{\boldsymbol{u}}, \widehat{\boldsymbol{u}}^{-1} \in \mathbb{C}^n$ can be calculated using $\mathcal{O}(n \ln n)$ operations.*

Exercise 3.8.2. *Finish the proof of Theorem 3.8.2 by arguing that $\widehat{\boldsymbol{u}}^{-1} \in \mathbb{C}^n$, like $\widehat{\boldsymbol{u}} \in \mathbb{C}^n$, can also always be calculated using $\mathcal{O}(n \ln n)$ operations. What changes need to be made to* (3.40) – (3.41)*?*

Theorem 3.8.2 generalizes Theorem 3.8.1 to handle all values of n efficiently. We are now in the position to declare that the DFT of any vector in $\mathbb{C}^n$ can be calculated using only $\mathcal{O}(n \ln n)$ operations! We are now prepared to prove that any (square) Toeplitz matrix has a fast matrix-vector multiplication algorithm.

3.8.2 Fast Matrix Multiplication for Toeplitz Matrices

Let $\mathbf{a} \in \mathbb{C}^{2n-1}$ and consider the $n \times n$ Toeplitz matrix generated by $\mathbf{a}$, $\text{Toep}_n(\mathbf{a}) \in \mathbb{C}^{n \times n}$. Given $\mathbf{v} \in \mathbb{C}^n$, we want to compute $\text{Toep}_n(\mathbf{a})\mathbf{v} \in \mathbb{C}^n$ using as few operations as possible.

Algorithm 7 Fast Toeplitz Matrix Multiplication

```
Input: $\mathbf{a} \in \mathbb{C}^{2n-1}$, $\mathbf{v} \in \mathbb{C}^n$
Output: $\mathrm{Toep}_n(\mathbf{a})\mathbf{v} \in \mathbb{C}^n$
$\mathbf{c} \leftarrow (a_{n-1}, a_{n-2}, \dots, a_0, 0, a_{2n-2}, \dots, a_n)^T \in \mathbb{C}^{2n}$
Compute $\widehat{\mathbf{c}} \in \mathbb{C}^{2n}$ using the FFT
Compute $\widehat{\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}} \in \mathbb{C}^{2n}$ using the FFT
$\widehat{\mathbf{b}} \leftarrow \sqrt{2n}\, \widehat{\mathbf{c}} \odot \widehat{\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}} \in \mathbb{C}^{2n}$
Compute $\mathbf{b} \in \mathbb{C}^{2n}$ using the IFFT
Return $(b_0, b_1, \dots, b_{n-1})^T \in \mathbb{C}^n$
```

Recalling Lemma 3.7.7 we can begin by embedding $\mathrm{Toep}_n(\mathbf{a})$ into a $2n \times 2n$ circulant matrix. Specifically, we have seen that the vector $\mathbf{c} = (a_{n-1}, a_{n-2}, \dots, a_0, 0, a_{2n-2}, \dots, a_n)^T \in \mathbb{C}^{2n}$ satisfies $(\mathrm{circ}(\mathbf{c}))_{j,k} = (\mathrm{Toep}_n)\,(\mathbf{a})_{j,k}$ for all $j, k \in [n]$. This then further implies that $(\mathrm{Toep}_n(\mathbf{a})\mathbf{v})_j = \left(\mathrm{circ}(\mathbf{c}) \begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}\right)_j$ for all $j \in [n]$ by (3.28). Hence, we can compute $\mathrm{Toep}_n(\mathbf{a})\mathbf{v}$ by computing the convolution $\mathbf{c} \star \begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}$. Finally, we further seen in (3.31) that this convolution can be computed efficiently via

$$\mathbf{c} \star \begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix} = \sqrt{2n}\; F^* \left(\widehat{\mathbf{c}} \odot \widehat{\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}}\right).$$

See Algorithm 7 for streamlined pseudocode.

Considering the runtime of Algorithm 7, we can see that forming both $\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}, \mathbf{c} \in \mathbb{C}^{2n}$ can be accomplished in $\mathcal{O}(n)$ time. In addition, the (entrywise) Hadamard product of any two vectors in $\mathbb{C}^{2n}$, as well as selecting the first n entries of any vector $\mathbf{b} \in \mathbb{C}^{2n}$, can also always be accomplished in $\mathcal{O}(n)$ time. Finally, each (I)FFT can be computed using $\mathcal{O}(n \log n)$ operations by Theorem 3.8.2. Hence, Algorithm 7 will always utilize a total of $\mathcal{O}(n \log n)$ operations in order to compute $\mathrm{Toep}_n(\mathbf{a})\mathbf{v} \in \mathbb{C}^n$. This is significantly faster than direct $\mathcal{O}(n^2)$-time matrix-vector multiplication when n is large.

Fast Matrix Multiplication for Rectangular Toeplitz Matrices

Note that Algorithm 7 only applies to *square* Toeplitz matrices. A very natural next question then becomes what we can do if we instead need to quickly multiply a large non-square (rectangular) Toeplitz matrix, $\mathrm{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m \times n}$, against a vector $\mathbf{v} \in \mathbb{C}^n$? One simple strategy for handling such problems involves re-expressing the rectangular Toeplitz matrix in a block-matrix form, where each resulting block is a smaller square Toeplitz submatrix. The large rectangular matrix $\mathrm{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m\times n}$ can then be multiplied against a given $\mathbf{v} \in \mathbb{C}^n$ by combining the results of its smaller square Toeplitz submatrices multiplied against (appropriate pieces of) $\mathbf{v}$, each of which can now be computed efficiently using, e.g., Algorithm 7.[12]

Example 3.8.3. *Let* $\mathbf{a} \in \mathbb{C}^6$ *and* $\mathbf{v} \in \mathbb{C}^2$. *Suppose that we want to compute* $\mathrm{Toep}_{5,2}(\mathbf{a})\mathbf{v} \in \mathbb{C}^5$. *Instead of computing the result directly we can instead, e.g., decompose* $\mathrm{Toep}_{5,2}(\mathbf{a})$ *into two* 2×2 *and two* 1×1 *Toeplitz submatrices, and then compute* $\mathrm{Toep}_{5,2}(\mathbf{a})\mathbf{v}$ *using the resulting block-matrix form via*

$$\mathrm{Toep}_{5,2}(\mathbf{a})\mathbf{v} = \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \\ a_2 & a_3 \\ a_1 & a_2 \\ a_0 & a_1 \end{pmatrix}\mathbf{v} = \begin{pmatrix} \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \end{pmatrix} \\ \\ \begin{pmatrix} a_2 & a_3 \\ a_1 & a_2 \end{pmatrix} \\ \\ \begin{pmatrix} a_0 \end{pmatrix} \; \begin{pmatrix} a_1 \end{pmatrix} \end{pmatrix}\mathbf{v} = \begin{pmatrix} \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \end{pmatrix}\mathbf{v} \\ \\ \begin{pmatrix} a_2 & a_3 \\ a_1 & a_2 \end{pmatrix}\mathbf{v} \\ \\ \begin{pmatrix} a_0 \end{pmatrix} v_0 + \begin{pmatrix} a_1 \end{pmatrix} v_1 \end{pmatrix}.$$

Note that the fact that $\mathrm{Toep}_{5,2}(\mathbf{a})$ *has constant diagonals ensures that all of its square submatrices above are also Toeplitz.*

Exercise 3.8.3. *Suppose that we are given* $\mathrm{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m\times n}$ *and integers* $1 \le p \le \min\{m, n\}$, $h \in [m-p+1]$, *and* $\ell \in [n-p+1]$. *Let* $A \in \mathbb{C}^{p\times p}$ *be such that* $A_{j,k} := \left(\mathrm{Toep}_{m,n}(\mathbf{a})\right)_{h+j,\ell+k}$ *for all* $j, k \in [p]$. *Show that* A *is Toeplitz.*

Exercise 3.8.4. *Let* $p, n \in \mathbb{N}$, $\mathrm{Toep}_{pn,n}(\mathbf{a}) \in \mathbb{C}^{pn\times n}$, *and* $\mathbf{v} \in \mathbb{C}^n$. *Show that* $\mathrm{Toep}_{pn,n}(\mathbf{a})\mathbf{v} \in \mathbb{C}^{pn}$ *can be computed in* $\mathcal{O}(pn\log n)$ *operations.*

Exercise 3.8.5. *Let* $q(x) = \sum_{j=0}^{n-1} q_j x^j$ *and* $r(x) = \sum_{j=0}^{n-1} r_j x^j$ *be two polynomials of degree at most* $n-1$. *Let* $t(x) = q(x)\cdot r(x)$ *be their product. We know that* $t(x)$ *is a polynomial of degree at most* $2n-2$ *which can be written as* $t(x) = \sum_{j=0}^{2n-2} t_j x^j$. *Write psuedocode for an algorithm that will compute the coefficients* t_j *of* $t(x)$ *in* $\mathcal{O}(n\ln n)$ *total operations.*

Exercise 3.8.6. *Let* $g : [0,1] \to \mathbb{R}$ *be a twice continuously differentiable and periodic function. Any such* g *will have a Fourier series expansion of the form*

$$g(x) = \sum_{\omega\in\mathbb{Z}} c_\omega \mathrm{e}^{2\pi \mathrm{i}\omega x} \quad \forall x \in [0,1],$$

where the Fourier series coefficients $c_\omega \in \mathbb{C}$ *satisfy (i)* $c_\omega = \overline{c_{-\omega}}$ *for all* $\omega \in \mathbb{Z}$, *and (ii)* $\sum_{\omega\in\mathbb{Z}} |c_\omega| < \infty$. *Let* $\boldsymbol{u} \in \mathbb{R}^n$ *be a vector whose entries are given by* $u_j = g(j/n)$ *for all*

[12] Of course, the best way to decompose a given $m\times n$ Toeplitz matrix into square Toeplitz submatrices depends on how m and n compare with one another.

$j \in [n]$. *Show that the vector* $F\boldsymbol{u} \in \mathbb{C}^n$ *has entries*

$$(F\boldsymbol{u})_j = \sqrt{n} \sum_{\omega \equiv j \text{ mod } n} c_\omega.$$

Rapidly Evaluating Convolutional Layers of Neurons

In addition to having fewer parameters than general layers of neurons, we can now see that convolutional layers of neurons (recall Definition 3.7.3) also have other computational advantages. Consider, e.g., a convolutional layer of neurons $\ell : \mathbb{R}^N \to \mathbb{R}^N$ defined by $\ell(\mathbf{x}) := \sigma\left(W\mathbf{x} + \mathbf{b}\right)$ with Toeplitz weight matrix $W = \text{Toep}_N(\mathbf{w}) \in \mathbb{R}^{N \times N}$. Evaluating $\ell(\mathbf{x})$ as part of a neural network forward-evaluation requires us to: (i) Compute $W\mathbf{x}$, (ii) add $\mathbf{b}$ to $W\mathbf{x}$ to compute $W\mathbf{x} + \mathbf{b}$, and then (iii) compute $\sigma\left(W\mathbf{x} + \mathbf{b}\right)$ by applying the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ to each entry of $W\mathbf{x} + \mathbf{b}$. Assuming that σ is a simple function, both steps (ii) and (iii) can always be accomplished in $\mathcal{O}(N)$ operations. The first step (i) therefore always dominates the layer's evaluation cost.

Focussing on step (i) above, we can see that it can be accomplished for $W = \text{Toep}_N(\mathbf{w})$ in $\mathcal{O}(N \log N)$-operations via Algorithm 7. Hence, evaluating $\ell(\mathbf{x})$ can also be accomplished in $\mathcal{O}(N \log N)$ total operations in this case. In contrast, a general layer of neurons must generally rely on direct $\mathcal{O}(N^2)$-time matrix-vector multiplication to complete step (i). Thus, convolutional layers of neurons require fewer operations to evaluate than general layers of neurons – yet another advantage of their Toeplitz structure!

3.9 ℓ^p-Norms and the Hölder Inequality

So far we have made extensive use of the ℓ^2-norm in nearly every previous section above. This is largely due to its relationship with its associated inner product, which, frankly, makes the ℓ^2-norm so heavenly that its difficult to imagine ever wanting to use anything else. However, despite its many magnificent properties, the ℓ^2-norm is not always the best choice in every situation. For this reason many other vector norms are also commonly considered in mathematical books and papers including, perhaps most commonly, other ℓ^p vector norms.

Definition 3.9.1 (The ℓ^p-norm on $\mathbb{C}^n$)**.** *Let* $p \in [1, \infty)$*. The* ***ℓ^p-norm of*** $\mathbf{x} \in \mathbb{C}^n$ *is*

$$\|\mathbf{x}\|_p := \left(\sum_{j \in [n]} |\mathbf{x}_j|^p \right)^{\frac{1}{p}}.$$

Exercise 3.9.1. *Let* $\alpha \in \mathbb{C}$ *and* $\mathbf{x} \in \mathbb{C}^n$*. Show that* (i) $\|\alpha \mathbf{x}\|_p = |\alpha| \|\mathbf{x}\|_p$*, and that* (ii) $\|\mathbf{x}\|_p = 0$ *if and only if* $\mathbf{x} = \mathbf{0}$*.*

Exercise 3.9.2. *Let* $\mathbf{x} \in \mathbb{C}^n$*. Show that* $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_p \leq n^{\frac{1}{p}}\|\mathbf{x}\|_\infty$ *holds for all* $p \in [1,\infty)$*. Conclude that* $\lim_{p\to\infty}\|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty$*.*

Looking at Definition 3.9.1 together with the last exercise, we can see that the ℓ^1, ℓ^2, and ℓ^∞ norms previously introduced in Section 2.2.3 are all effectively special cases of the ℓ^p vector norms considered here. Hence, it is perhaps not very surprising that, as in these special cases, establishing the triangle inequality for general ℓ^p vector norms is more difficult than establishing the other two norm properties (see Exercise 3.9.1). Much of the remainder of this section will be spent proving the triangle inequality for general ℓ^p vector norms (also known as the Minkowski inequality) as a result. Following the approach in [29], we will now briefly discuss convex functions before proceeding with the proof.

3.9.1 Convex Functions of One Variable

The following definition is crucial below.

Definition 3.9.2 (Convex Functions from $\mathbb{R}$ into $\mathbb{R}$)**.** *A function* $f : \mathbb{R} \to \mathbb{R}$ *is* **convex on** (a,b) *if and only if*

$$f\left(tx + (1-t)y\right) \leq tf(x) + (1-t)f(y) \tag{3.42}$$

holds $\forall t \in [0,1]$ *and* $x, y \in (a,b)$*.*

Note that (3.42) always holds when $t = 0$, $t = 1$, or $x = y$. Hence, it suffices to only consider $t \in (0,1)$ and $x \neq y$ below.

Definition 3.9.3 (Concave Functions from $\mathbb{R}$ into $\mathbb{R}$)**.** *A function* $f : \mathbb{R} \to \mathbb{R}$ *is* **concave on** (a,b) *if* $(-f)(x) := -f(x)$ *is convex on* (a,b)*.*

Exercise 3.9.3. *Let* $m, b \in \mathbb{R}$*. Show that the linear function* $f(x) := mx + b$ *is both concave and convex on* $(-\infty,\infty)$*.*

The next lemma provides a useful alternate characterization of convexity.

Lemma 3.9.4. *The function* $f : \mathbb{R} \to \mathbb{R}$ *is convex on* (a,b) *if and only if* $\forall x_1, x_2, x_3 \in (a,b)$ *with* $x_1 < x_2 < x_3$ *we have*

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} \leq \frac{f(x_3) - f(x_2)}{x_3 - x_2}.$$

Proof. Let $t \in (0,1)$ and suppose, without loss of generality, that the $x, y \in (a,b)$ in (3.42) are such that $x < y$. Set $x_1 = x$, $x_2 = tx + (1-t)y$, and $x_3 = y$. Note that in this case we have that $x_2 - x_1 = (t-1)x_1 + (1-t)x_3 = (1-t)(x_3 - x_1)$ and $x_3 - x_2 = t(x_3 - x_1)$ so that

$$t = \frac{x_3 - x_2}{x_3 - x_1}, \ (1-t) = \frac{x_2 - x_1}{x_3 - x_1}, \text{ and } \frac{1-t}{t} = \frac{x_2 - x_1}{x_3 - x_2}.$$

Now we can see that f is convex on (a, b) if and only if

$$\begin{aligned} f(x_2) \ \leq \ tf(x_1) + (1-t)f(x_3) \quad &\Longleftrightarrow \quad t\,(f(x_2) - f(x_1)) \leq (1-t)\,(f(x_3) - f(x_2)) \\ &\Longleftrightarrow \quad f(x_2) - f(x_1) \leq \frac{1-t}{t}\,(f(x_3) - f(x_2)) \\ &\Longleftrightarrow \quad f(x_2) - f(x_1) \leq \left(\frac{x_2 - x_1}{x_3 - x_2}\right)(f(x_3) - f(x_2)). \end{aligned}$$

Dividing both sides of the last inequality above by $x_2 - x_1$ proves the result. □

Using Lemma 3.9.4 together with the Mean Value Theorem from calculus (see, e.g., [46, Theorem 7.20]) we can now prove that functions $f : \mathbb{R} \to \mathbb{R}$ with non-decreasing derivatives on $(a, b) \subset \mathbb{R}$ are also convex on (a, b).

Theorem 3.9.5 (Mean Value Theorem)**.** *Suppose that $f : \mathbb{R} \to \mathbb{R}$ is continuous on $[a, b]$ and differentiable on (a, b). Then $\exists c \in (a, b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Theorem 3.9.6 (Functions with Non-Decreasing Derivatives are Convex)**.** *Suppose that $f : \mathbb{R} \to \mathbb{R}$ has a non-decreasing derivative f' on (a, b). Then, f is convex on (a, b).*

Proof. Choose any $x_1 < x_2 < x_3$ you like in (a, b). By Theorem 3.9.5 $\exists c_1 \in (x_1, x_2)$ and $c_2 \in (x_2, x_3)$ such that

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = f'(c_1) \ \leq \ f'(c_2) = \frac{f(x_3) - f(x_2)}{x_3 - x_2}.$$

The result now follows from Lemma 3.9.4. □

Exercise 3.9.4. *Suppose that $f : \mathbb{R} \to \mathbb{R}$ has a non-negative second derivative f'' on (a, b) so that $f''(x) \geq 0$ for all $x \in (a, b)$. Show that f is convex on (a, b). HINT: Use the fundamental theorem of calculus.*

Exercise 3.9.5. *Let $p \geq 1$. Show that $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^p$ is convex on $(0, \infty)$.*

Exercise 3.9.6. *Show that the natural logarithm, $\log : (0, \infty) \to \mathbb{R}$, is concave on $(0, \infty)$.*

We are now equipped to prove a basic result about the convexity of an important function of one complex variable. It will be crucial to our proof of the Minkowski inequality.

Lemma 3.9.7. *Let $p \geq 1$. Then*

$$|tz + (1-t)w|^p \leq t|z|^p + (1-t)|w|^p$$

holds for all $t \in [0, 1]$ and $z, w \in \mathbb{C}$.

Proof. The result holds if either z or w is 0, so assume that both are nonzero. Let $g : \mathbb{R} \to \mathbb{R}$ be $g(x) = x^p$ and note that g is both non-decreasing, and convex on $(0, \infty)$ by Exercise 3.9.5. Using the properties of the magnitude function $|\cdot| : \mathbb{C} \to \mathbb{R}^+$ (2.5) together with the fact that g is non-decreasing we have that

$$|tz + (1-t)w| \leq |tz| + |(1-t)w| = t|z| + (1-t)|w|$$

which implies that

$$g\left(|tz + (1-t)w|\right) \;=\; |tz + (1-t)w|^p \;\leq\; g\left(t|z| + (1-t)|w|\right) \;=\; \left(t|z| + (1-t)|w|\right)^p.$$

Hence, since g is convex on $(0, \infty)$, we also have that

$$|tz + (1-t)w|^p \;\leq\; (t|z| + (1-t)|w|)^p \;\leq\; t|z|^p + (1-t)|w|^p$$

as we wished to show. □

We are now finally able to prove the Minkowski inequality.

3.9.2 The Minkowski Inequality for Vectors

We are now able to prove the triangle inequality for the ℓ^p-norm on $\mathbb{C}^n$, thereby concluding our proof that it is indeed a vector norm.

Theorem 3.9.8 (Minkowski Inequality)**.** *Let $p \geq 1$. Then*

$$\|\mathbf{x} + \mathbf{y}\|_p \leq \|\mathbf{x}\|_p + \|\mathbf{y}\|_p$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.

Proof. Note that the result trivially holds if either $\mathbf{x}$ or $\mathbf{y}$ is $\mathbf{0}$, so we assume both are nonzero such that $c := \|\mathbf{x}\|_p + \|\mathbf{y}\|_p > 0$. Let $\tilde{\mathbf{x}} := \mathbf{x}/c$ and $\tilde{\mathbf{y}} := \mathbf{y}/c$. By our choice of c we have that $\|\tilde{\mathbf{x}}\|_p =: t$ and $\|\tilde{\mathbf{y}}\|_p = 1 - t$ for some $t \in (0, 1)$. Now letting $\tilde{\mathbf{x}}' := \tilde{\mathbf{x}}/\|\tilde{\mathbf{x}}\|_p$ and $\tilde{\mathbf{y}}' := \tilde{\mathbf{y}}/\|\tilde{\mathbf{y}}\|_p$ we can see that

$$\begin{aligned} \|\mathbf{x} + \mathbf{y}\|_p \leq \|\mathbf{x}\|_p + \|\mathbf{y}\|_p \quad &\Longleftrightarrow \quad \|t\tilde{\mathbf{x}}' + (1-t)\tilde{\mathbf{y}}'\|_p \leq 1 \\ &\Longleftrightarrow \quad \|t\tilde{\mathbf{x}}' + (1-t)\tilde{\mathbf{y}}'\|_p^p \leq 1. \end{aligned}$$

We may now use Lemma 3.9.7 to verify the last inequality above by seeing that

$$\begin{aligned} \|t\tilde{\mathbf{x}}' + (1-t)\tilde{\mathbf{y}}'\|_p^p \;&=\; \sum_{j \in [n]} \left|t\tilde{x}_j' + (1-t)\tilde{y}_j'\right|^p \;\leq\; \sum_{j \in [n]} t\left|\tilde{x}_j'\right|^p + (1-t)\left|\tilde{y}_j'\right|^p \\ &=\; t\|\tilde{\mathbf{x}}'\|_p^p + (1-t)\|\tilde{\mathbf{y}}'\|_p^p \;=\; 1. \end{aligned}$$

The result follows. □

It is often useful to use easily computable ℓ^p-norms to help control other ℓ^q-norms. We will develop a crucial tool for obtaining related bounds in the next section.

Exercise 3.9.7. *Let $q \geq p \geq 1$. Show that $\|\mathbf{x}\|_q \leq \|\mathbf{x}\|_p$.*
HINT: Let $\mathbf{y} := \mathbf{x}/\|\mathbf{x}\|_q$ and then prove that $\|\mathbf{y}\|_q = 1 \leq \|\mathbf{y}\|_p$.

3.9.3 Young's Inequality for Products & the Discrete Hölder Inequality

We will begin with a very useful result about good-ole-fashioned real numbers.

Lemma 3.9.9 (Young's Inequality for Products)**.** *Let $p, q \in (1, \infty)$ satisfy $\frac{1}{p} + \frac{1}{q} = 1$. Then,*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}$$

holds for all $a, b \in [0, \infty)$.

Proof. The equality clearly holds if either a or b is 0, so we'll assume that both are positive. Since $\log : (0, \infty) \to \mathbb{R}$ is concave on $(0, \infty)$ by Exercise 3.9.6, we have that

$$\begin{aligned} -\log\left(\frac{1}{p}a^p + \frac{1}{q}b^q\right) &= -\log\left(\frac{1}{p}a^p + \left(1 - \frac{1}{p}\right)b^q\right) \leq -\frac{1}{p}\log\left(a^p\right) - \left(1 - \frac{1}{p}\right)\log\left(b^q\right) \\ &= -\log(a) - \log(b) = -\log(ab). \end{aligned}$$

Hence, $\log(ab) \leq \log\left(\frac{1}{p}a^p + \frac{1}{q}b^q\right)$. Exponentiating both sides of this last inequality now proves the result. □

We now have all the tools we need to prove a generalized version of the Cauchy–Schwarz inequality.

Theorem 3.9.10 (The Hölder Inequality)**.** *Let $p, q \in (1, \infty)$ satisfy $\frac{1}{p} + \frac{1}{q} = 1$. Then,*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \sum_{j \in [n]} |\overline{x_j}\, y_j| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q$$

holds $\forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.

Proof. The first inequality always holds by properties of the magnitude function $|\cdot| : \mathbb{C} \to \mathbb{R}^+$ (2.5) (check!). Thus, it suffices to show that $\sum_{j \in [n]} |\overline{x_j}\, y_j| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q$.

Let $\tilde{\mathbf{x}} := \frac{\mathbf{x}}{\|\mathbf{x}\|_p}$ and $\tilde{\mathbf{y}} := \frac{\mathbf{y}}{\|\mathbf{y}\|_q}$. Then by Lemma 3.9.9 we have that

$$\begin{aligned} \sum_{j \in [n]} \frac{|\overline{x_j}\, y_j|}{\|\mathbf{x}\|_p \|\mathbf{y}\|_q} &= \sum_{j \in [n]} |\tilde{x}_j|\, |\tilde{y}_j| \leq \sum_{j \in [n]} \frac{|\tilde{x}_j|^p}{p} + \frac{|\tilde{y}_j|^q}{q} \\ &= \frac{\|\tilde{\mathbf{x}}\|_p^p}{p} + \frac{\|\tilde{\mathbf{y}}\|_q^q}{q} = \frac{1}{p} + \frac{1}{q} = 1. \end{aligned}$$

Multiplying both sides of the inequality above by $\|\mathbf{x}\|_p \|\mathbf{y}\|_q$ now finishes the proof. □

Exercise 3.9.8. *Let $\alpha \in [0, \infty)$ and $p, q \in (1, \infty)$ satisfy $\frac{1}{p} + \frac{1}{q} = 1$. Suppose that $\mathbf{x}, \mathbf{y} \in [0, \infty)^n$ have $x_j = \alpha\, y_j^{q/p} / \|\mathbf{y}\|_q^{q-1}$ $\forall j \in [n]$. Prove that $\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\|_p \|\mathbf{y}\|_q$ holds. <u>HINT</u>: Begin by showing that $\|\mathbf{x}\|_p = \alpha$.*

Exercise 3.9.9. *Prove that* $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_\infty \|\mathbf{y}\|_1$ *also holds* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.

Given that both Exercise 3.9.9 and Theorem 3.9.10 hold, we will apply the Hölder Inequality for all $p, q \in [1, \infty]$ satisfying $\frac{1}{p} + \frac{1}{q} = 1$ below (i.e., we will implicitly define $1/\infty := 0$).

Exercise 3.9.10. *Prove that* $\|\mathbf{x}\|_2 \leq \sqrt{\|\mathbf{x}\|_\infty \|\mathbf{x}\|_1}$ *holds* $\forall \mathbf{x} \in \mathbb{C}^n$.

Exercise 3.9.11. *Let* $p \geq 1$ *and* $r, q \in (1, \infty)$ *satisfy* $\frac{1}{r} + \frac{1}{q} = 1$. *Prove that* $\|\mathbf{x}\|_{2p} \leq \sqrt{\|\mathbf{x}\|_{pq} \|\mathbf{x}\|_{pr}}$ *holds* $\forall \mathbf{x} \in \mathbb{C}^n$.

3.10 Some Discrete Inequalities from Fourier Analysis

In this section we will briefly discuss the Riesz–Thorin interpolation theorem in the context of finite dimensional linear algebra along with a few of its most important implications for DFT (3.29) and circulant matrices. This section roughly follows the same path as the corresponding portions in [22, Chapter 8], [30, Chapter 4], and [3, Chapter 1], except restricted to the finite dimensional setting. The interested reader is referred to these sources for more details and discussion, as well as to [1] for a clever application of these results to a dimensionality reduction problem in the computer science literature.

To begin our journey we will now generalize the (ℓ^2, ℓ^2)-operator norm introduced in Section 2.2.3.

Definition 3.10.1. *Let* $p, q \in [1, \infty]$ *and* $A \in \mathbb{C}^{m \times n}$. *The* (ℓ^p, ℓ^q)***-operator norm of*** A *is*

$$\|A\|_{p \to q} := \max_{\mathbf{x} \in \mathbb{C}^n \text{ with } \|\mathbf{x}\|_p = 1} \|A\mathbf{x}\|_q = \sup_{\mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}} \frac{\|A\mathbf{x}\|_q}{\|\mathbf{x}\|_p}.$$

As with the (ℓ^2, ℓ^2)-operator norm, one of the main uses of the (ℓ^p, ℓ^q)-operator norm of a matrix A is to help bound the ℓ^q-norm of $A\mathbf{x}$ when one has knowledge of about the ℓ^p-norm of $\mathbf{x}$. The following lemma summarizes this standard use of operator norms.

Lemma 3.10.2. *Let* $A \in \mathbb{C}^{m \times n}$ *and* $\mathbf{x} \in \mathbb{C}^n$. *Then* $\|A\mathbf{x}\|_q \leq \|A\|_{p \to q} \|\mathbf{x}\|_p$ *holds for all* $p, q \in [1, \infty]$.

Proof. If $\mathbf{x} = \mathbf{0}$ we're done, so assume that $\mathbf{x} \neq \mathbf{0}$. Then,

$$\|A\mathbf{x}\|_q = \left(\frac{\|A\mathbf{x}\|_q}{\|\mathbf{x}\|_p} \right) \|\mathbf{x}\|_p \leq \left(\sup_{\mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}} \frac{\|A\mathbf{x}\|_q}{\|\mathbf{x}\|_p} \right) \|\mathbf{x}\|_p = \|A\|_{p \to q} \|\mathbf{x}\|_p.$$

□

Exercise 3.10.1. *Let* $A \in \mathbb{C}^{m \times n}$ *and* $B \in \mathbb{C}^{n \times p}$. *Prove that* $\|AB\|_{p \to p} \leq \|A\|_{p \to p} \|B\|_{p \to p}$ *holds for all* $p \geq 1$.

Exercise 3.10.2. *Let $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times p}$. Prove that $\|AB\|_{p\to q} \leq \|A\|_{r\to q}\|B\|_{p\to r}$ holds for all $p, q, r \geq 1$.*

Exercise 3.10.3. *Let $p \in [1,\infty]$ and $A \in \mathbb{C}^{m\times n}$. Use Exercise 3.9.8 to help you prove that*

$$\|A\|_{p\to p} \;=\; \sup_{\mathbf{x}\in\mathbb{C}^n\setminus\{\mathbf{0}\},\ \mathbf{y}\in\mathbb{C}^m\setminus\{\mathbf{0}\}} \frac{\langle \mathbf{y}, A\mathbf{x}\rangle}{\|\mathbf{y}\|_q\|\mathbf{x}\|_p},$$

where $q \in [1,\infty]$ satisfies $1/q + 1/p = 1$.

Exercise 3.10.4. *Let $p \in [1,\infty]$ and $A \in \mathbb{C}^{m\times n}$. Show that $\|A\|_{p\to p} = \|A^*\|_{q\to q}$, where $q \in [1,\infty]$ satisfies $1/q + 1/p = 1$.*

The next lemma and exercise demonstrate that some matrix operator norms are in fact quite easy to compute.

Lemma 3.10.3. *Let $A \in \mathbb{C}^{m\times n}$. Then $\|A\|_{1\to1} = \max_{k\in[n]} \sum_{j\in[m]} |A_{j,k}|$.*

Proof. Note that

$$\begin{aligned}\|A\|_{1\to1} \;&=\; \max_{\mathbf{x}\in\mathbb{C}^n \text{ with } \|\mathbf{x}\|_1=1} \|A\mathbf{x}\|_1 \;=\; \max_{\mathbf{x}\in\mathbb{C}^n \text{ with } \|\mathbf{x}\|_1=1} \sum_{j\in[m]} \left|\sum_{k\in[n]} A_{j,k}\, x_k\right| \\ &\leq\; \max_{\mathbf{x}\in\mathbb{C}^n \text{ with } \|\mathbf{x}\|_1=1} \sum_{k\in[n]} |x_k| \left(\sum_{j\in[m]} |A_{j,k}|\right) \;\leq\; \max_{k\in[n]} \sum_{j\in[m]} |A_{j,k}|.\end{aligned}$$

Furthermore, this upper bound is achieved since $\|A\mathbf{e}_k\|_1 \;=\; \|A_{:,k}\|_1$ holds for all $k \in [n]$. □

Exercise 3.10.5. *Let $A \in \mathbb{C}^{m\times n}$. Prove that $\|A\|_{\infty\to\infty} = \max_{j\in[m]} \sum_{k\in[n]} |A_{j,k}|$.*

Exercise 3.10.6. *Let $F \in \mathbb{C}^{n\times n}$ be the DFT matrix of size n. Show that $\|F\|_{2\to2} = 1$ and $\|F\|_{1\to\infty} = 1/\sqrt{n}$.*

Theorem 3.10.4 (Discrete Riesz–Thorin (see, e.g., Theorem 1.1.1 in [3]))**.** *Let $A \in \mathbb{C}^{m\times n}$, $\theta \in (0,1)$, and $p, q, p_1, q_1, p_2, q_2 \in [1,\infty]$ be such that $p_1 \neq p_2$, $q_1 \neq q_2$, $1/p = \theta/p_1 + (1-\theta)/p_2$, and $1/q = \theta/q_1 + (1-\theta)/q_2$. Then,*

$$\|A\|_{p\to q} \;\leq\; \left(\|A\|_{p_1\to q_1}\right)^{\theta} \left(\|A\|_{p_2\to q_2}\right)^{1-\theta}.$$

Using the Riesz–Thorin Theorem we can now prove some additional useful operator norm bounds for F. Choose $\theta \in (0,1)$ and suppose that $1/p \;=\; \theta/1 + (1-\theta)/2 \;=\; \frac{1+\theta}{2}$ and $1/q \;=\; \theta/\infty + (1-\theta)/2 \;=\; \frac{1-\theta}{2}$. Summing these expressions we can see that $1/p + 1/q = 1$, and also that $\theta = 1 - \frac{2}{q}$. Applying Riesz–Thorin together with Exercise 3.10.6 we now have

$$\|F\|_{p\to q} \;\leq\; \left(\|F\|_{1\to\infty}\right)^{1-\frac{2}{q}} \left(\|F\|_{2\to2}\right)^{\frac{2}{q}} \;=\; n^{\frac{1}{q}-\frac{1}{2}} \;=\; n^{\frac{1}{2}-\frac{1}{p}}.$$

Finally, noting that $p \in (1,2)$ since $\theta \in (0,1)$ we obtain the following generalization of Exercise 3.10.6.

Theorem 3.10.5 (Discrete Hausdorff-Young)**.** *Let $p \in [1,2]$ and $q = \frac{p}{p-1}$ (with $1/0 := \infty$). Then $\|F\|_{p\to q} \ \le \ n^{\frac{1}{2}-\frac{1}{p}}$.*

Exercise 3.10.7. *Let $\mathbf{v} \in \mathbb{C}^n$. Prove that $\|\mathrm{circ}(\mathbf{v})\|_{2\to 2} \ = \ \sqrt{n}\|F\mathbf{v}\|_\infty \ = \ \sqrt{n}\cdot\max_{\omega\in[n]} |\widehat{v}_\omega|$ and that $\|\mathrm{circ}(\mathbf{v})\|_{1\to 1} \ = \ \|\mathrm{circ}(\mathbf{v})\|_{\infty\to\infty} \ = \ \|\mathbf{v}\|_1$.*

Exercise 3.10.8. *Let $\mathbf{v} \in \mathbb{C}^n$. Prove that $\|\mathrm{circ}(\mathbf{v})\|_{1\to\infty} \le \|\mathbf{v}\|_\infty$.*

Exercise 3.10.9. *Let $\mathbf{v} \in \mathbb{C}^n$. Prove that*

$$\|\mathrm{circ}(\mathbf{v})\|_{p\to p} \le \begin{cases} \|\mathbf{v}\|_1^{2/p-1}\left(\sqrt{n}\|F\mathbf{v}\|_\infty\right)^{2-\frac{2}{p}} & \forall p\in[1,2] \\ \|\mathbf{v}\|_1^{1-\frac{2}{p}}\left(\sqrt{n}\|F\mathbf{v}\|_\infty\right)^{2/p} & p \ge 2 \end{cases}.$$

Exercise 3.10.10. *Prove that $\|F\mathbf{v}\|_\infty \ \le \ \|\mathbf{v}\|_1/\sqrt{n}$ holds $\forall \mathbf{v} \ \in \ \mathbb{C}^n$. Conclude that $\|\mathrm{circ}(\mathbf{v})\|_{p\to p} \le \|\mathbf{v}\|_1$ holds $\forall \mathbf{v} \in \mathbb{C}^n$.*

3.10.1 The Discrete Young's Convolution Inequality

In this section we will provide some useful bounds on the norm of the convolutions of two vectors. Our first result (which you will prove in the next exercise) is a simple consequence of Exercise 3.10.10.

Exercise 3.10.11. *Prove that $\|\mathbf{u}\star\mathbf{v}\|_p \le \min\{\|\mathbf{u}\|_1\|\mathbf{v}\|_p,\ \|\mathbf{v}\|_1\|\mathbf{u}\|_p\}$ holds for all $p \ge 1$ and $\mathbf{u},\mathbf{v} \in \mathbb{C}^n$.*

Our next result (which you will prove just below) is a consequence of Hölder's inequality.

Exercise 3.10.12. *Let $p, q \in [1,\infty]$ be such that $1/p + 1/q = 1$. Show that $\|\mathbf{u}\star\mathbf{v}\|_\infty \le \min\{\|\mathbf{u}\|_p\|\mathbf{v}\|_q,\ \|\mathbf{v}\|_p\|\mathbf{u}\|_q\}$ holds for all $\mathbf{u},\mathbf{v} \in \mathbb{C}^n$.*

Now let $\mathbf{v} \in \mathbb{C}^n$ and $p, q \in (1,\infty)$ be such that $\frac{1}{p}+\frac{1}{q} = 1$. From Exercise 3.10.11 we have that $\|\mathrm{circ}(\mathbf{v})\|_{1\to p} \le \|\mathbf{v}\|_p$, and from Exercise 3.10.12 we also have that $\|\mathrm{circ}(\mathbf{v})\|_{q\to\infty} \le \|\mathbf{v}\|_p$. Choose $\theta \in (0,1)$ and suppose that

$$\frac{1}{r} \ = \ \frac{\theta}{1} + \frac{1-\theta}{q}, \text{ and that } \frac{1}{s} \ = \ \frac{\theta}{p} + \frac{1-\theta}{\infty} \ = \ \frac{\theta}{p}.$$

Applying Theorem 3.10.4 tells us that

$$\|\mathrm{circ}(\mathbf{v})\|_{r\to s} \ \le \ \left(\|\mathrm{circ}(\mathbf{v})\|_{1\to p}\right)^\theta \left(\|\mathrm{circ}(\mathbf{v})\|_{q\to\infty}\right)^{1-\theta} \ = \ \|\mathbf{v}\|_p.$$

Note that since $\frac{1}{r} \in (1/q, 1)$ we can see that $r \in (1,q)$, and

$$\frac{1}{r} \ = \ \theta + (1-\theta)\frac{1}{q} \ = \ \theta + \frac{1}{q} - \frac{\theta}{q} \ = \ \theta + \frac{1}{q} - \theta\left(1 - \frac{1}{p}\right) \ = \ \frac{1}{q} + \frac{\theta}{p} \ = \ \frac{1}{q} + \frac{1}{s}.$$

The following result follows from the discussion and exercises above.

Theorem 3.10.6 (Discrete Young's Convolution Inequality)**.** *Let $\mathbf{v} \in \mathbb{C}^n$ and $p, q \in [1, \infty]$ be such that $\frac{1}{p} + \frac{1}{q} = 1$. Pick $r \in [1, q]$ and set $\frac{1}{s} = \frac{1}{r} - \frac{1}{q}$. Then, $\|\text{circ}(\mathbf{v})\|_{r \to s} \leq \|\mathbf{v}\|_p$.*

Exercise 3.10.13. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$ and $p, q \in [1, \infty]$ be such that $1/p + 1/q = 1$. Show that $\|\mathbf{u} \star \mathbf{v}\|_{(1/r - 1/q)^{-1}} \leq \min\{\|\mathbf{u}\|_p \|\mathbf{v}\|_r,\ \|\mathbf{v}\|_p \|\mathbf{u}\|_r\}$ holds for all $r \in [1, q]$.*

3.11 Embedding Metric Spaces into Normed Vector Spaces

In this section we will briefly discuss how linear algebra can also begin to help with very general computational tasks that don't even explicitly involve any vectors. This section largely follows [35]. We refer the interested reader there, as well as to, e.g., [26]. To begin we will need to recall the notion of a "finite metric space".

Definition 3.11.1. *A* **metric space** *is a pair $(\mathcal{S}, \rho)$ where $\mathcal{S}$ is a set, and $\rho : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ is a function (called a* **metric***) that satisfies the following three properties:*

1. *$\rho(a, b) = 0$ if and only if $a = b$,*

2. *$\rho(a, b) \ = \ \rho(b, a)$ $\forall a, b \in \mathcal{S}$, and*

3. *$\rho(a, c) \ \leq \ \rho(a, b) + \rho(b, c)$ $\forall a, b, c \in \mathcal{S}$.*

A **finite metric space** *is a metric space where $|\mathcal{S}|$ is finite.*

Exercise 3.11.1. *Let $\mathcal{S} \subset \mathbb{C}^n$ and $\rho : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ be $\rho(\mathbf{x}, \mathbf{y}) \ = \ \|\mathbf{x} - \mathbf{y}\|$, where $\|\cdot\|$ is a norm on $\mathbb{C}^n$. Prove that $(\mathcal{S}, \rho)$ is a metric space.*

Example 3.11.2 (Weighted Graph Metric Spaces)**.** *Let $(\mathcal{V}, \mathcal{E})$ be a simple and connected weighted graph with vertex set $\mathcal{V}$, edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and weight function $w : E \to (0, \infty)$. A* **path of length** *d* **from vertex** *$u \in \mathcal{V}$* **to** *$v \in \mathcal{V}$ is a subset of edges $\mathcal{P} = \{e_0, \dots, e_{d-1}\} \subset E$ such that (i) $e_j = (p, q) \in \mathcal{P}$ if and only if $e_{j-1} = (\cdot, p) \in \mathcal{P}$ and $e_{j+1} = (q, \cdot) \in \mathcal{P}$ for all $j = 1, \dots, d-2$, (ii) $e_0 = (u, \cdot) \in \mathcal{P}$ and, (iii) $e_d = (\cdot, v) \in \mathcal{P}$. That is, a path $\mathcal{P}$ of length d from u to v is a string of d edges that connects u and v. Note that since $(\mathcal{V}, \mathcal{E})$ is connected we are guaranteed that there is at least one path from every vertex to every other vertex.*

One can show that the shortest path distance from $u \in \mathcal{V}$ to $v \in \mathcal{V}$ defined by

$$\rho(u, v) := \begin{cases} \inf\limits_{\textit{paths } \mathcal{P} \textit{ from } u \textit{ to } v} \displaystyle\sum_{e_j \in \mathcal{P}} w(e_j) & \text{if } u \neq v \\ 0 & \text{if } u = v \end{cases}.$$

is a metric on $\mathcal{V}$. See Figure 3.3 for an example.

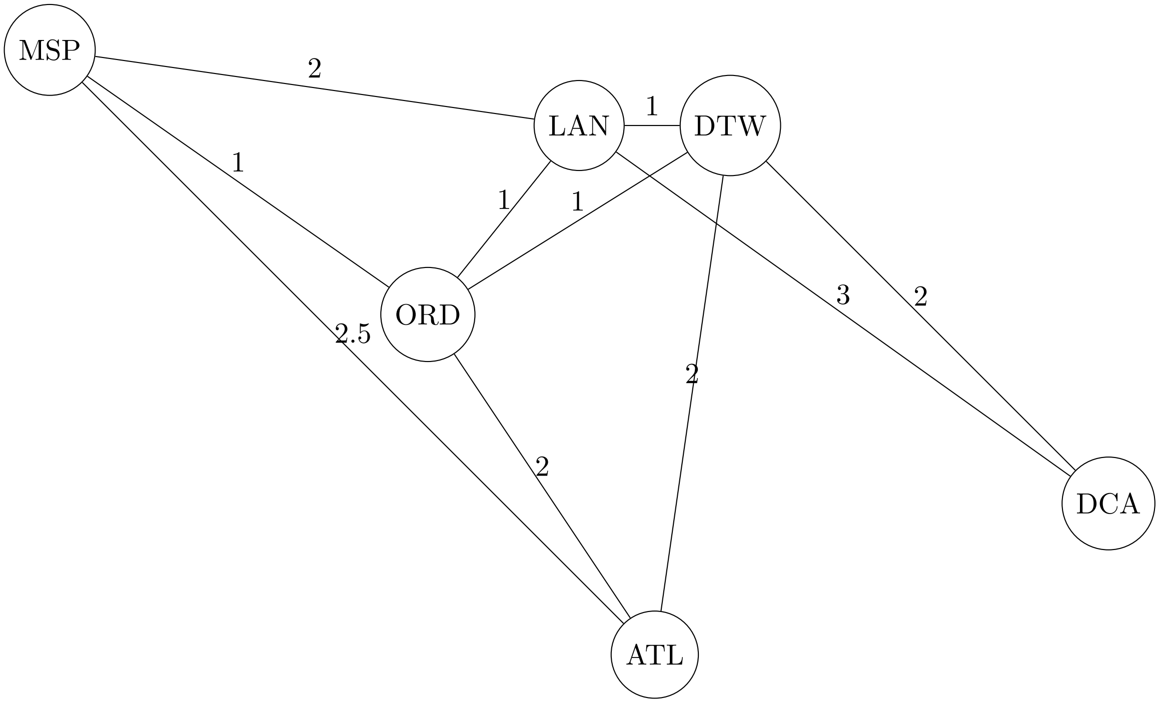

Figure 3.3: Example of a weighted graph using airport codes as the nodes. Here $\mathcal{V} = \{\text{ATL}, \text{DCA}, \text{DTW}, \text{LAN}, \text{MSP}, \text{ORD}\}$ and the edge set $\mathcal{E}$ includes, e.g., $\{(\text{DCA}, \text{DTW}), (\text{DTW}, \text{DCA}), (\text{DCA}, \text{LAN}), (\text{LAN}, \text{DCA})\}$ as a subset of its 18 total elements. The edges represent available flights over a certain period of time, and the numerical weights, e.g., $w((\text{DTW}, \text{DCA})) = 2 = w((\text{DCA}, \text{DTW}))$, represent travel time. The path $\mathcal{P} = \{(\text{LAN}, \text{DTW}), (\text{DTW}, \text{ATL})\}$ is a shortest path from LAN to ATL. Its distance is 3.

Suppose that we are given an extremely large and potentially complicated finite metric space $(\mathcal{V}, \rho)$ in the form of, e.g., a much larger graph along the lines of Figure 3.3. Further imagine that we are going to need to rapidly compute many distances, $\rho(u, v)$, between many different and unpredictable user-generated query pairs $(u, v) \in \mathcal{V} \times \mathcal{V}$ over a long period of time. Finally, suppose also that we are willing to sacrifice some accuracy in our computed value for each $\rho(u, v)$-query in exchange for using less memory/time to compute its value (i.e., we are OK with an answer that's *approximately correct* as long as it's also a lot cheaper to compute).[13] In such a setting the mission now becomes the following: we wish

[13] Indeed, why not accept an approximately correct answer? As we hope to have communicated already in, e.g., Section 3.7.3, every answer computed on a digital computer is only approximately correct anyway.... Embrace the mess, and remember with humility that our mathematics can never be more than a helpful

to determine the best possible tradeoff we can between the accuracy of our approximately computed $\rho(u,v)$-distances and the time/memory it takes to compute them. In keeping with the chapter so far, our strategy for exploring this tradeoff will involve using ℓ^p-norms to help us approximately compute ρ-distances more cheaply via "embeddings".

Definition 3.11.3. *Let $0 < \alpha \leq \beta$, and $q \in [1,\infty]$. We say that a metric space $(\mathcal{S},\rho)$ is* (α,β)**-embeddable into** ℓ^q_n *if there exists a function $f : \mathcal{S} \to \mathbb{R}^n$ such that*

$$\alpha\rho(u,v) \leq \|f(u) - f(v)\|_q \leq \beta\rho(u,v) \quad \forall u,v \in \mathcal{S}.$$

Here $f : \mathcal{S} \to \mathbb{R}^n$ is called an (α,β)**-embedding of** $(\mathcal{S},\rho)$ **into** ℓ^q_n. *If $\alpha = \beta = 1$ above we say that* $(\mathcal{S},\rho)$ **embeds isometrically into** ℓ^q_n, *and call $f : \mathcal{S} \to \mathbb{R}^n$ an* **isometric embedding of** $(\mathcal{S},\rho)$ **into** ℓ^q_n.

Exercise 3.11.2. *Let $0 < \alpha \leq \beta$, and $q \in [1,\infty]$. Suppose that $f : \mathcal{S} \to \mathbb{R}^n$ is an (α,β)-embedding of a finite metric space $(\mathcal{S},\rho)$ into ℓ^q_n. Prove that $|f(\mathcal{S})| = |\mathcal{S}|$.*

Our next theorem is both instructive and a bit silly. It effectively tells us that storing all possible distances between elements of a finite metric space ensures that we can then exactly re-compute them all again later using the ℓ^∞-norm. Of course, if we have already stored all possible distances, then "re-computing them" thereafter should be trivial – we can just read them back. Hence the silliness.... That said, we can't actually just read a given individual distance back – we have to somehow extract it using the ℓ^∞-norm. Seeing how/why this can be done is instructive.

Theorem 3.11.4. *Every finite metric space $(\mathcal{S},\rho)$ embeds isometrically into $\ell^\infty_{|\mathcal{S}|}$.*

Proof. Given $\mathcal{S} = \{u_j\}_{j\in[|\mathcal{S}|]}$ define $f : \mathcal{S} \to \mathbb{R}^{|\mathcal{S}|}$ by

$$f(u_j) := \left(\rho(u_0,u_j), \rho(u_1,u_j)\ldots, \rho(u_{|\mathcal{S}|-1},u_j)\right)^T \in [0,\infty)^{|\mathcal{S}|}.$$

Considering the infinity norm of the vectors in $f(\mathcal{S}) - f(\mathcal{S})$ we can see that

$$\|f(u_j) - f(u_k)\|_\infty \geq \left|(f(u_j) - f(u_k))_k\right| = |\rho(u_k,u_j) - \rho(u_k,u_k)| = \rho(u_k,u_j)$$

holds for all $j,k \in [|\mathcal{S}|]$. Hence, $\alpha = 1$ in Definition 3.11.3.

On the other hand,

$$\left|(f(u_j) - f(u_k))_l\right| = |\rho(u_l,u_j) - \rho(u_l,u_k)| \leq \rho(u_j,u_k) = \rho(u_k,u_j)$$

holds for all $l \in [|\mathcal{S}|]$ by the (reverse) triangle inequality. As a result, $\|f(u_j) - f(u_k)\|_\infty \leq \rho(u_j,u_k)$ also holds for all $j,k \in [|\mathcal{S}|]$. Hence, $\beta = 1$ in Definition 3.11.3 as well, proving the desired result. □

ghost pointing the general direction through a very corporeal silicon minefield.

Looking at the proof of Theorem 3.11.4 we can see that we are simply storing all distances between all pairs of elements of $\mathcal{S}$ in a large symmetric (i.e., Hermitian) matrix whose columns define the isometric embedding $f : \mathcal{S} \to \mathbb{R}^{|\mathcal{S}|}$.

$$\begin{array}{cc}
\begin{array}{ccccc} & & f(u_j) & & \\ & & \downarrow & & \end{array} & \\
\left(\begin{array}{cc|c|cc}
\cline{3-3}
\rho(u_0, u_0) & \cdots & \rho(u_0, u_j) & \cdots & \rho(u_0, u_{|\mathcal{S}|-1}) \\
\rho(u_1, u_0) & \cdots & \rho(u_1, u_j) & \cdots & \rho(u_1, u_{|\mathcal{S}|-1}) \\
\vdots & & \vdots & & \vdots \\
\rho(u_{|\mathcal{S}|-1}, u_0) & \cdots & \rho(u_{|\mathcal{S}|-1}, u_j) & \cdots & \rho(u_{|\mathcal{S}|-1}, u_{|\mathcal{S}|-1}) \\
\cline{3-3}
\end{array}\right) & \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}
\end{array}$$

Given this, the isometric embedding provided by Theorem 3.11.4 doesn't really provide any computational benefits on its own. If you're already going to store all $\mathcal{O}(|\mathcal{S}|^2)$-possible pairwise distances $\rho(u_j, u_k)$, then computing them later should be as easy as a $\mathcal{O}(1)$-time memory access using your favorite implementation of a hash table.

That valid point made, however, the story begins to change if the Hermitian matrix above is (approximately) low rank. In such cases one can imagine beginning to save on the memory needed to store the matrix above by using, e.g., ideas from Section 3.4. If this type of approach sounds appealing, we invite the interested reader to explore the closely related world of spectral graph theory [41]. If you've survived this far you certainly have all of the linear algebraic preliminaries you need to begin. Before potentially taking this detour, however, we encourage the reader to first look at the next section where an isometric embedding similar to the one provided by Theorem 3.11.4 does in fact end up making an interesting class of computations faster.

3.11.1 Rapidly Approximating the Diameter of a Set of Vectors

Consider the **ℓ^p-diameter** of a set $\mathcal{S} \subset \mathbb{R}^n$, defined by

$$\mathrm{diam}_p(\mathcal{S}) := \sup_{\mathbf{x},\mathbf{y}\in\mathcal{S}} \|\mathbf{x} - \mathbf{y}\|_p.$$

Computing the ℓ^p-diameter of a finite set $\mathcal{S}$ directly by simply calculating the ℓ^p-norm of all $\binom{|\mathcal{S}|}{2}$ pairs of vectors in $\mathcal{S}$ takes $\mathcal{O}(n|\mathcal{S}|^2)$-operations. This can quickly become costly when $|\mathcal{S}|$ is very large.

Now imagine that n is extremely very small compared to $|\mathcal{S}|$ (e.g., that $n = 3$ as is the case in LiDAR point cloud data [11]). In this case it's potentially worthwhile to effectively increase n in exchange for reducing the $|\mathcal{S}|^2$-scaling of the direct $\mathrm{diam}_p(\mathcal{S})$ calculation. In this section we will present an embedding-based method for realizing exactly this type of tradeoff between n and $|\mathcal{S}|$. The following lemma will be crucial to our approach.

Lemma 3.11.5. *Let $\mathcal{S} \subset \mathbb{R}^n$ be finite and let $\rho(\mathbf{x},\mathbf{y}) := \|\mathbf{x}-\mathbf{y}\|_1$. Then, the finite metric space $(\mathcal{S},\rho)$ embeds isometrically into $\ell^\infty_{2^n}$ via a linear isometric embedding $f : \mathcal{S} \to \mathbb{R}^{2^n}$.*

Proof. Let $\tilde{\mathcal{S}} \subset \mathbb{R}^{2^n}$ be the set $\{-1,1\}^n$ of all 2^n possible n-length vectors of ±1s. Recalling that $\text{sign} : \mathbb{R} \to \{1,-1\}$ is

$$\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

we have that

$$\|\mathbf{x}\|_1 = \sum_{j=1}^{n} |x_j| = \sum_{j=1}^{n} \text{sign}(x_j)x_j \leq \max_{\mathbf{y}\in\tilde{\mathcal{S}}} \langle \mathbf{y},\mathbf{x}\rangle.$$

On the other hand, using Hölder's inequality we can also see that

$$\max_{\mathbf{y}\in\tilde{\mathcal{S}}} \langle \mathbf{y},\mathbf{x}\rangle \leq \|\mathbf{y}\|_\infty \|\mathbf{x}\|_1 = \|\mathbf{x}\|_1.$$

Hence, $\|\mathbf{x}\|_1 = \max_{\mathbf{y}\in\tilde{\mathcal{S}}} \langle \mathbf{y},\mathbf{x}\rangle$. We can now define our linear function $f : \mathbb{R}^n \to \mathbb{R}^{2^n}$ to be such that

$$f(\mathbf{x}) = (\langle \mathbf{x},\mathbf{y}\rangle)_{\mathbf{y}\in\tilde{\mathcal{S}}}.$$

From above we have that $\|\mathbf{x}\|_1 = \|f(\mathbf{x})\|_\infty$ so that f is indeed an isometric embedding. □

The next exercise will also be useful.

Exercise 3.11.3. *Let $p \geq 1$. Prove that*

$$\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_1 \leq \|\mathbf{x}\|_p \; n^{1-1/p}$$

holds $\forall \mathbf{x} \in \mathbb{R}^n$. HINT: Apply Exercise 3.9.7 together with Hölder's Inequality.

The computational utility of Lemma 3.11.5 comes from the fact that the number of operations required to compute $\text{diam}_\infty(\mathcal{S})$ only scales linearly in $|\mathcal{S}|$. We can see this by considering the following calculation. Note that for any $\mathcal{S} \subset \mathbb{R}^m$ we have

$$\begin{aligned} \text{diam}_\infty(\mathcal{S}) &= \max_{\mathbf{x},\mathbf{y}\in\mathcal{S}} \|\mathbf{x}-\mathbf{y}\|_\infty = \max_{\mathbf{x},\mathbf{y}\in\mathcal{S}} \max_{j\in[m]} |x_j - y_j| = \max_{j\in[m]} \max_{\mathbf{x},\mathbf{y}\in\mathcal{S}} |x_j - y_j| \\ &= \max_{j\in[m]} \left| \max_{\mathbf{x}\in S} x_j - \min_{\mathbf{y}\in\mathcal{S}} y_j \right| = \max_{j\in[m]} \left(\max_{\mathbf{x}\in S} x_j - \min_{\mathbf{y}\in\mathcal{S}} y_j \right). \end{aligned} \tag{3.43}$$

Above, computing $\max_{\mathbf{x}\in\mathcal{S}} x_j$ and $\min_{\mathbf{y}\in\mathcal{S}} y_j$ both require $\mathcal{O}(|\mathcal{S}|)$ comparisons for each $j \in [m]$. Thus, computing $\text{diam}_\infty(\mathcal{S})$ takes just $\mathcal{O}(m|\mathcal{S}|)$ total operations (i.e., roughly as many as it takes to simply read $\mathcal{S}$ into memory). This observation motivates Algorithm 8 for rapidly approximating the ℓ^p-diameter of a given set $\mathcal{S} \subset \mathbb{R}^n$.

Considering the computational cost of Algorithm 8, we recall that $f(\mathbf{x})$ from Lemma 3.11.5 is computed by taking 2^n inner products of vectors of length n. Since this is computed

Algorithm 8 Rapidly Approximate the ℓ^p-Diameter of $\mathcal{S} \subset \mathbb{R}^n$

```
1:  Input: $\mathcal{S} \subset \mathbb{R}^n,\ p \in [1, \infty)$.
2:  Output: Estimate of $\mathbf{diam}_p(\mathcal{S})$.
    # Apply the isometric embedding $f : \mathbb{R}^n \to \mathbb{R}^{2^n}$ from Lemma 3.11.5.
3:  Initialize $\mathcal{S}' = \{\}$.
4:  for $\mathbf{x} \in \mathcal{S}$ do
5:      Compute $\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^{2^n}$ where $f$ is defined as in Lemma 3.11.5.
6:      Let $\mathcal{S}' \ = \ \mathcal{S}' \cup \{\mathbf{y}\}$.
7:  end for
    # Compute $diam_\infty(\mathcal{S}')$ as per (3.43).
8:  for each $j \in [2^n]$ do
9:      Let $\max_j = \max_{\mathbf{y} \in \mathcal{S}'} y_j$.
10:     Let $\min_j = \min_{\mathbf{y} \in \mathcal{S}'} y_j$.
11:     Let $\mathrm{diff}_j = \max_j - \min_j$.
12: end for
13: return $\max_{j \in [2^n]} \mathrm{diff}_j$.
```

for each element of $\mathbf{x} \in \mathcal{S}$, the first loop of Algorithm 8 (i.e.,m lines 3 – 7) is $(n 2^n |\mathcal{S}|)$. The second loop (lines 8 – 12) requires comparing $\mathcal{O}(|\mathcal{S}|)$ values for each of the 2^n coordinates for a total of $\mathcal{O}(2^n|\mathcal{S}|)$-operations. Thus, the overall runtime of Algorithm 8 is $\mathcal{O}(n2^n|\mathcal{S}|)$. In contrast, recall that the directly computing $\mathrm{diam}_p(\mathcal{S})$ uses $\mathcal{O}(n|\mathcal{S}|^2)$-operations. Hence, we can expect that Algorithm 8 will be faster than directly computing $\mathrm{diam}_p(\mathcal{S})$ whenever $|\mathcal{S}|$ is much greater than 2^n.

Note, however, that approximating $\mathrm{diam}_p(\mathcal{S})$ quickly is only impressive if our approximation is actually accurate. Speed without accuracy means nothing. If we only cared about speed we could, for example, simply output a random number as our approximation to $\mathrm{diam}_p(\mathcal{S})$ every time – that would be faster than anything else we have discussed so far (only $\mathcal{O}(1)$-time!). Of course, such random guessing approach would also provide incredibly terrible answers just about every time we ran it, making it practically worthless....[14] Thankfully, we can show that Algorithm 8 is always guaranteed to be pretty accurate, especially when n is relatively small.

Theorem 3.11.6. *Let $p \geq 1$. Algorithm 8 always outputs an estimate $e \in \mathbb{R}$ satisfying*

$$diam_p(\mathcal{S}) \ \leq \ e \ \leq \ n^{1-\frac{1}{p}} \ diam_p(\mathcal{S}).$$

Proof. By Lemma 3.11.5 we have that the isometric embedding $f : \mathbb{R}^n \to \mathbb{R}^{2^n}$ satisfies

$$e \ := \ \max_{\mathbf{x},\mathbf{y} \in \mathcal{S}} \|f(\mathbf{x}) - f(\mathbf{y})\|_\infty \ = \ \max_{\mathbf{x},\mathbf{y} \in \mathcal{S}} \|\mathbf{x} - \mathbf{y}\|_1.$$

[14] On the other hand, even a broken clock tells the correct time twice a day... :).

The result now follows from Exercise 3.11.3 since $\|\mathbf{x}-\mathbf{y}\|_p \leq \|\mathbf{x}-\mathbf{y}\|_1 \leq n^{1-\frac{1}{p}}\|\mathbf{x}-\mathbf{y}\|_p$ holds for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}$. □

Having gotten a little more experience with embedding methods in this last section, we will now return to the question of whether Theorem 3.11.4 can be improved to use less memory (i.e., to compute fewer ρ-distances).

3.11.2 Fréchet Embedding Methods for Finite Metric Spaces

We will begin by defining the functional structure that determines a Fréchet embedding.

Definition 3.11.7 (Fréchet Function)**.** *Let $(\mathcal{S}, \rho)$ be a finite metric space and $\mathcal{S}_j \subset \mathcal{S}$ be a non-empty subset of $\mathcal{S}$ for all $j \in [n]$. Choose $\gamma_0, \dots, \gamma_{n-1} \in \mathbb{R}^+$. A Fréchet function $f : \mathcal{S} \to \mathbb{R}^n$ is a function whose j^{th} component function $f_j : \mathcal{S} \to \mathbb{R}^+$ is*

$$(f(u))_j \;=\; f_j(u) \;:=\; \gamma_j \min_{v \in \mathcal{S}_j} \rho(u,v) \;\; \forall u \in \mathcal{S}, j \in [n]. \tag{3.44}$$

The following lemma shows that a large class of Fréchet functions will automatically have a well-behaved β-parameter as an (α, β)-embedding of $(\mathcal{S}, \rho)$ into ℓ_n^q.

Lemma 3.11.8. *Let $q \in [1, \infty)$, $(\mathcal{S}, \rho)$ be a finite metric space, and $f : \mathcal{S} \to \mathbb{R}^n$ be a Fréchet function as in Definition 3.11.7 with $\gamma_0 = \gamma_1 = \cdots = \gamma_{n-1} \;=:\; \gamma$. Then*

$$\|f(u) - f(v)\|_q \leq \gamma n^{1/q} \rho(u,v)$$

holds $\forall u, v \in \mathcal{S}$.

Proof. Fix $u, v \in \mathcal{S}$. We will show that for any nonempty subset $\mathcal{T} \subset \mathcal{S}$ we have

$$|\rho(u,\mathcal{T}) - \rho(v,\mathcal{T})| \leq \rho(u,v) \tag{3.45}$$

where $\rho(u,\mathcal{T}) := \min_{w \in \mathcal{T}} \rho(u,w)$. If (3.45) holds then

$$\left(\sum_{j=1}^{n} |\gamma \rho(u,\mathcal{S}_j) - \gamma \rho(v,\mathcal{S}_j)|^q \right)^{1/q} \leq \gamma \left(\sum_{j=1}^{n} |\rho(u,v)|^q \right)^{1/q} = \gamma n^{1/q} \rho(u,v).$$

To see that (3.45) holds for nonempty $\mathcal{T} \subset \mathcal{S}$ note that $\rho(u,\mathcal{T}) \leq \rho(u,w) \;\forall w \in \mathcal{T}$ by definition. Thus, if $\tilde{w} \in \mathcal{T}$ satisfies $\rho(v,\mathcal{T}) = \rho(v,\tilde{w})$ we can see that

$$\rho(u,\mathcal{T}) - \rho(v,\mathcal{T}) \leq \rho(u,\tilde{w}) - \rho(v,\tilde{w}) \leq \rho(u,v).$$

Similarly, $\rho(v,\mathcal{T}) - \rho(u,\mathcal{T}) \leq \rho(u,v)$ holds when $\tilde{w}$ is chosen so that $\rho(u,\tilde{w}) = \rho(u,\mathcal{T})$. □

Exercise 3.11.4. *Consider the setup in Lemma 3.11.8. Prove that $\|f(u) - f(v)\|_\infty \leq \gamma \rho(u,v)$ also holds $\forall u, v \in \mathcal{S}$.*

Exercise 3.11.5. *Consider the setup in Lemma 3.11.8. Further suppose that the $\mathcal{S}_j$ subsets in Definition 3.11.7 satisfy the following separating condition: for all $(u, v) \in \mathcal{S} \times \mathcal{S} \setminus \{u\}$ there exists $j \in [n]$ such that $u \in \mathcal{S}_j$ and $v \notin \mathcal{S}_j$. Prove that $\exists \alpha > 0$ such that $\|f(u) - f(v)\|_q \geq \alpha \rho(u, v)$ for all $u, v \in \mathcal{S}$.*

Let $(\mathcal{S}, \rho)$ be a finite metric space. Lemma 3.11.8 shows that injective Fréchet functions with $\gamma_j = \gamma \ \forall j \in [n]$ are always (α, β)-embeddings of $(\mathcal{S}, \rho)$ into ℓ^q_n with $\beta = \gamma n^{1/q}$. The main difficulty is in choosing the $\mathcal{S}_j$ subsets in Definition 3.11.7 so that α can also be bounded away from zero in a nontrivial fashion. Ideally, we would like to have $\alpha = c\beta$ for some fixed constant c as close to 1 as possible while simultaneously reducing the number of ρ-distances we need to store/compute. As we will describe next, it turns out that simply choosing the $\mathcal{S}_j$ subsets in Definition 3.11.7 randomly can lead to computationally efficient (α, β)-embeddings that also have near-optimal $c = \alpha/\beta$ values.

Let $s = |\mathcal{S}|$, $q \in [2, \infty) \cap \mathbb{N}$, $p = s^{-1/q}$, and define the probabilities $p_j := \min\left\{\frac{1}{2},\ p^j\right\}$ for $j = 1, \dots, q$. Next, let $m = \left\lceil 24 s^{1/q} \log s \right\rceil$, and for each $j = 1, \dots, q$ and $k = 1, \dots, m$ form a random subset $\mathcal{S}_{j,k} \subset \mathcal{S}$ by independently including each $u \in \mathcal{S}$ in $\mathcal{S}_{j,k}$ with probability p_j. Finally, form a Fréchet function $f : \mathcal{S} \to \mathbb{R}^{qm}$ as per Definition 3.11.7 with $\gamma_0, \dots, \gamma_{qm-1} = 1$ using all $n = qm$ of these independently random $\mathcal{S}_{j,k}$-sets. The next theorem guarantees that this Fréchet function will in fact be an $\left(\frac{1}{2q-1}, 1\right)$-embedding of $(\mathcal{S}, \rho)$ into ℓ^∞_{qm} with high probability. See [35, Theorem 15.7.2] and [34] for details.

Theorem 3.11.9. *Let $q \geq 2$ be an integer and $(\mathcal{S}, \rho)$ be a finite metric space. There exists a Fréchet embedding $f : \mathcal{S} \to \mathbb{R}^n$ satisfying*

$$\frac{\rho(u, v)}{2q - 1} \leq \|f(u) - f(v)\|_\infty \leq \rho(u, v) \quad \forall u, v \in \mathcal{S}, \tag{3.46}$$

where $n = \mathcal{O}\left(q|\mathcal{S}|^{1/q} \log |\mathcal{S}|\right)$.

Suppose that we use the Fréchet embedding $f : \mathcal{S} \to \mathbb{R}^{\mathcal{O}\left(q|\mathcal{S}|^{1/q} \log |\mathcal{S}|\right)}$ promised by Theorem 3.11.9 to encode every element of $\mathcal{S}$ so that we can then rapidly approximate $\rho(u, v)$ for all $u, v \in \mathcal{S}$ thereafter using only the stored vectors $\{f(u)\}_{u \in \mathcal{S}} \subset \mathbb{R}^{\mathcal{O}\left(q|\mathcal{S}|^{1/q} \log |\mathcal{S}|\right)}$. This will require us to store $\mathcal{O}\left(q|\mathcal{S}|^{1+1/q} \log |\mathcal{S}|\right)$ total values. Comparing this to the $\mathcal{O}(|\mathcal{S}|^2)$ pairwise distances Theorem 3.11.4 stores, we can see that we can now effectively decrease the total number of values we need to store in exchange for allowing some approximation error in (3.46). It's also worth mentioning that general purpose data structures have been developed that allow for more rapid computation of the type of nearest-neighbor distances, $\min_{v \in \mathcal{S}_j} \rho(u, v)$, utilized by Fréchet functions (see, e.g., [4]). Such data structures can help make the evaluation of Fréchet embeddings quite efficient, allowing for even more computational gains. Finally, though we have focused on (α, β)-embeddings into ℓ^∞_n here, it's also interesting to note that similar randomized Fréchet embeddings into, e.g., ℓ^2_n also exist [6].

As a very final flourish to this chapter we'd like to celebrate the appearance of a new character just above: probability. Recall that proving Theorem 3.11.9 requires finding "good" $\mathcal{S}_j$ subsets to use in Definition 3.11.7. After noting this challenging design problem we almost immediately decided to randomly guess the $\mathcal{S}_j$ subsets to use (which is easy to do!), and then saw that doing so produced very nice results. This is just one example of a phenomena that occurs over and over and over again: probability can be used to make near-optimal design choices that are (often) simply too hard for people to figure out how to make any other way. We point this out mainly to motivate you to follow this recommendation: go learn some probability!!! Once you know even a little probability we are confident that it'll come in so handy so often that, looking back later, you won't know how you ever survived without it.

Bibliography

[1] N. Ailon and E. Liberty. Fast dimension reduction using rademacher series on dual bch codes. *Discrete & Computational Geometry*, 42(4):615–630, 2009.

[2] J. V. Atanasoff. Advent of electronic digital computing. *Annals of the History of Computing*, 6(3):229–282, 2008.

[3] J. Bergh and J. Löfström. *Interpolation spaces: an introduction*, volume 223. Springer Science & Business Media, 2012.

[4] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 97–104, New York, NY, USA, 2006. Association for Computing Machinery.

[5] L. I. Bluestein. A Linear Filtering Approach to the Computation of Discrete Fourier Transform. *IEEE Transactions on Audio and Electroacoustics*, 18:451–455, 1970.

[6] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

[7] J. P. Boyd. *Chebyshev and Fourier Spectral Methods.* Dover Publications, Inc., 2001.

[8] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*, pages 707–720. Springer, 2002.

[9] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[10] J. W. Brown and R. V. Churchill. *Complex variables and applications.* McGraw-Hill,, 2009.

[11] J. Carter, K. Schmid, K. Waters, L. Betzhold, B. Hadley, R. Mataosky, and J. Halleran. Lidar 101: An introduction to lidar technology, data, and applications. *National*

Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, `https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf`, 2012.

[12] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[13] B. A. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2, 2000.

[14] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19:297–301, 1965.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms. *2nd Edition*, 2001.

[16] J. W. Demmel. *Applied numerical linear algebra.* SIAM, 1997.

[17] S. Foucart. *Mathematical pictures at a data science exhibition.* Cambridge University Press, 2022.

[18] S. H. Friedberg, A. J. Insel, and L. E. Spence. *Linear Algebra (Fourth Edition).* Pearson Higher Ed, 2003.

[19] M. Frigo and S. Johnson. The design and implementation of fftw3. *Proceedings of IEEE 93 (2)*, pages 216–231, 2005.

[20] S. R. Garcia and R. A. Horn. *Matrix Mathematics: A Second Course in Linear Algebra.* Cambridge University Press, 2023.

[21] V. Guillemin and A. Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.

[22] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities.* Cambridge university press, 1952.

[23] M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the fast fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, 1984.

[24] R. Horn and C. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[25] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge university press, 2013.

[26] P. Indyk, J. Matouvsek, and A. Sidiropoulos. Low-distortion embeddings of finite metric spaces. In *Handbook of discrete and computational geometry*, pages 211–231. Chapman and Hall/CRC, 2017.

[27] M. A. Iwen and B. Ong. A distributed and incremental svd algorithm for agglomerative data analysis on large networks. *SIAM Journal on Matrix Analysis and Applications*, 37(4):1699–1718, 2016.

[28] M. A. Iwen and C. V. Spencer. A note on compressed sensing and the complexity of matrix multiplication. *Information Processing Letters*, 109(10):468–471, 2009.

[29] R. Kantrowitz and M. M. Neumann. Yet another proof of minkowski's inequality. *The American Mathematical Monthly*, 115(5):445–447, 2008.

[30] Y. Katznelson. *An introduction to harmonic analysis.* Cambridge University Press, 2004.

[31] D. R. Kincaid and E. W. Cheney. *Numerical analysis: mathematics of scientific computing*, volume 2. Brooks/Cole Pacific Grove, CA, 2002.

[32] R. Kress. *Numerical Analysis*, volume 181. Springer-Verlag, 1998.

[33] J. E. Marsden and A. Tromba. *Vector Calculus, Sixth Edition.* W.H. Freeman and Company, 2012.

[34] J. Matouvsek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996.

[35] J. Matouvsek. Embedding finite metric spaces into normed spaces. In *Lectures on Discrete Geometry*, pages 355–400. Springer, 2002.

[36] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An introduction to the theory of numbers.* John Wiley & Sons, 1991.

[37] C. H. Papadimitriou. *Computational Complexity.* Addiaon-Wesley Publishing Company, Inc., 1994.

[38] G. Plonka, D. Potts, G. Steidl, and M. Tasche. *Numerical Fourier Analysis.* Springer, 2018.

[39] L. Rabiner, R. Schafer, and C. Rader. The Chirp z-Transform Algorithm. *IEEE Transactions on Audio and Electroacoustics*, AU-17(2):86–92, June 1969.

[40] T. Shifrin and M. Adams. *Linear algebra: A geometric approach (Second Edition).* Macmillan, 2011.

[41] D. A. Spielman. Spectral and algebraic graph theory. `http://cs-www.cs.yale.edu/homes/spielman/sagt/`, 2025.

[42] G. Stewart. Perturbation theory for the singular value decomposition. *Digital Repository at the University of Maryland, UMIACS-TR-90-124 CS-TR 2539*, September 1990.

[43] G. Stewart and J.-g. Sun. *Matrix Perturbation Theory.* Computer Science and Scientific Computing/Academic Press, Inc, 1990.

[44] G. W. Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[45] V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[46] B. S. Thomson, J. B. Bruckner, and A. M. Bruckner. *Elementary Real Analysis.* Prentice Hall (Pearson), 2001.

[47] L. N. Trefethen and D. Bau. *Numerical linear algebra.* SIAM, 2022.

[48] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898, 2012.