A MATHEMATICAL INTRODUCTION TO FAST AND MEMORY EFFICIENT ALGORITHMS FOR BIG DATA

Mark Iwen

This is Version 0.1 – December 16, 2024: This document replaces version 0 which was initially posted on January 2, 2021. Many proofs are *still* very sketchy (pun intended), typos abound, referencing is incomplete, and some topics are not completely finished. Other topics have been excluded altogether to reduce the scope of these notes to a more manageable level. Please email me at will with complaints, questions, and suggestions. All feedback is welcome. Most Importantly, thanks to all my Michigan State students in MTH 994 and CMSE 890 in Fall 2020 and in MTH 800 in Fall 2024 for providing feedback on these notes. In particular, Chapters 1 and 2 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. Chapters 3 and 4 and many portions of other chapters were initially scribed by Cullen Haselby in the Fall Semester of 2020. And, large portions of Chapter 5 were initially scribed by Graig Gross in 2020.

M.A.I.

Contents

Chapter 1

Why We Should Care: Online Computing, 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 20^{th} 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, how one might be trained, 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 $N-1$, $[N] \coloneqq \{0, \dots, N-1\}$ 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 $f : \mathbb{R}^{3\times 10^6} \to \mathbb{R}^{3\times 10^6}$ 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*

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

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.

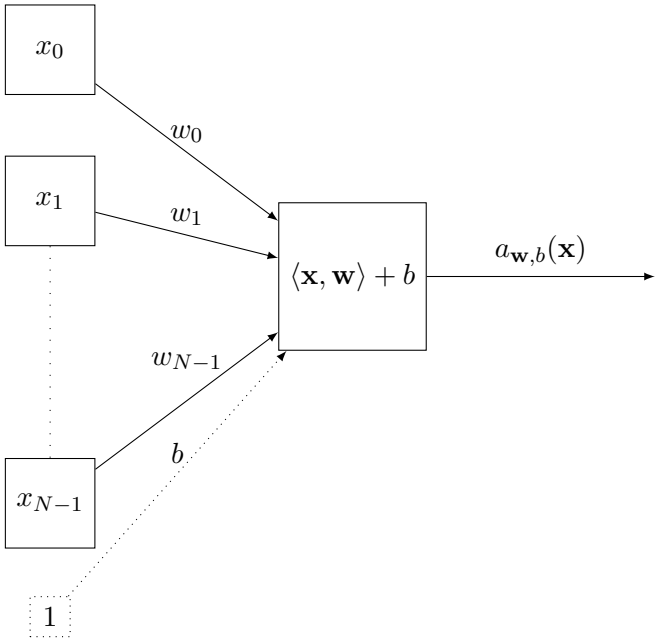


Figure 1.1: A graphical representation of an affine function $a_{\mathbf{w},b} : \mathbb{R}^N \to \mathbb{R}$. The first column of boxes represents the inputs (i.e., the entries of $\mathbf{x}$). The edge weights are the entry of the weight vector $\mathbf{w}$ that multiplies each corresponding input 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: $\sigma(y) = \tanh(y)$

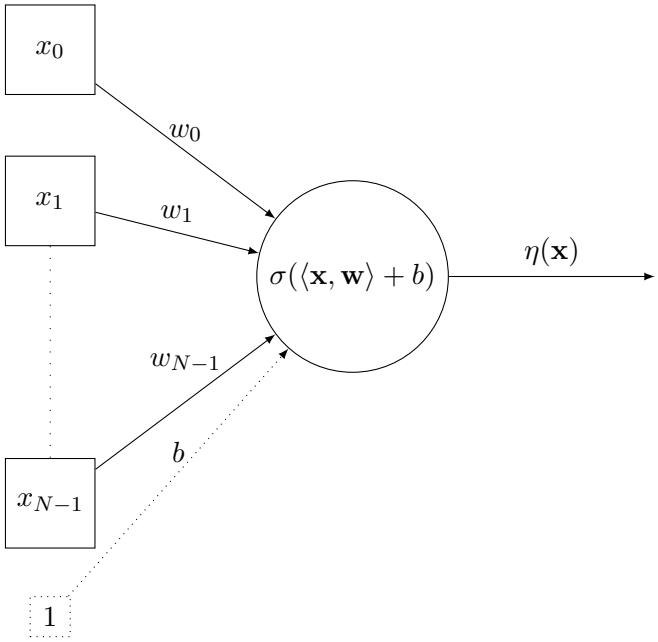


Figure 1.2: A graphical representation of a neuron. The first column of boxes represents the inputs (i.e., the entries of $\mathbf{x}$). The edge weights are the entry of the weight vector $\mathbf{w}$ that multiplies each corresponding input entry in the neuron's inner product (e.g., w_0 multiplies 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

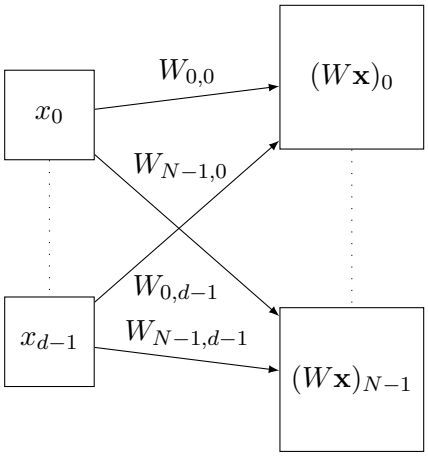


Figure 1.3: A graphical representation of a matrix $W \in \mathbb{R}^{N\times d}$ as an input layer of width d connected directly to a linear output layer of width N.

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 $\mathbf{w}_0, \ldots, \mathbf{w}_{d-1} \in \mathbb{R}^N$, d biases $b_0, \ldots, b_{d-1}$, 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* $\eta_0, \ldots, \eta_{d-1} : \mathbb{R}^N \to \mathbb{R}$ *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.

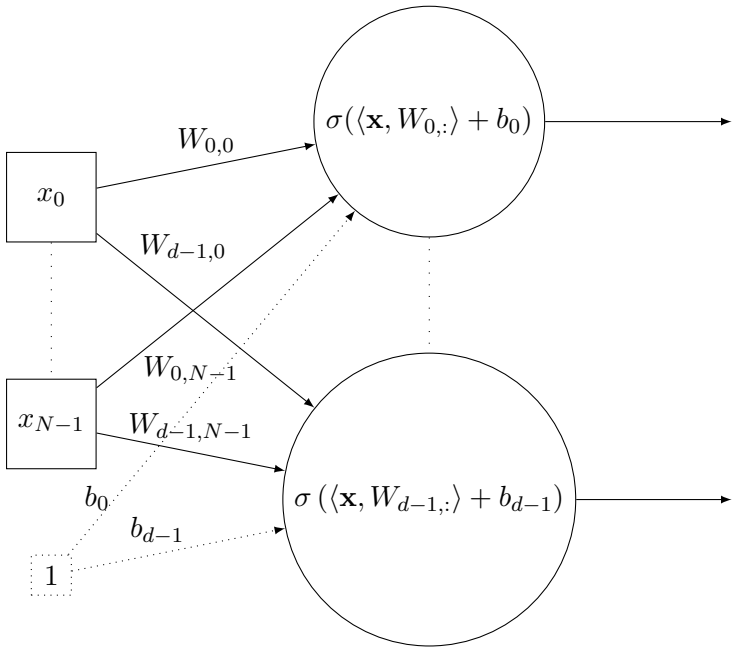


Figure 1.4: A graphical representation of a layer of neurons $\ell : \mathbb{R}^N \to \mathbb{R}^d$ defined by $\ell(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$ with weight matrix $W \in \mathbb{R}^{d\times N}$ and bias vector $\mathbf{b} \in \mathbb{R}^d$. Here the input layer of width N connects 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 2.2.5 and 2.3. 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_j^0 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 $w_j^0, w_j^1 \in \mathbb{R}$ 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.*

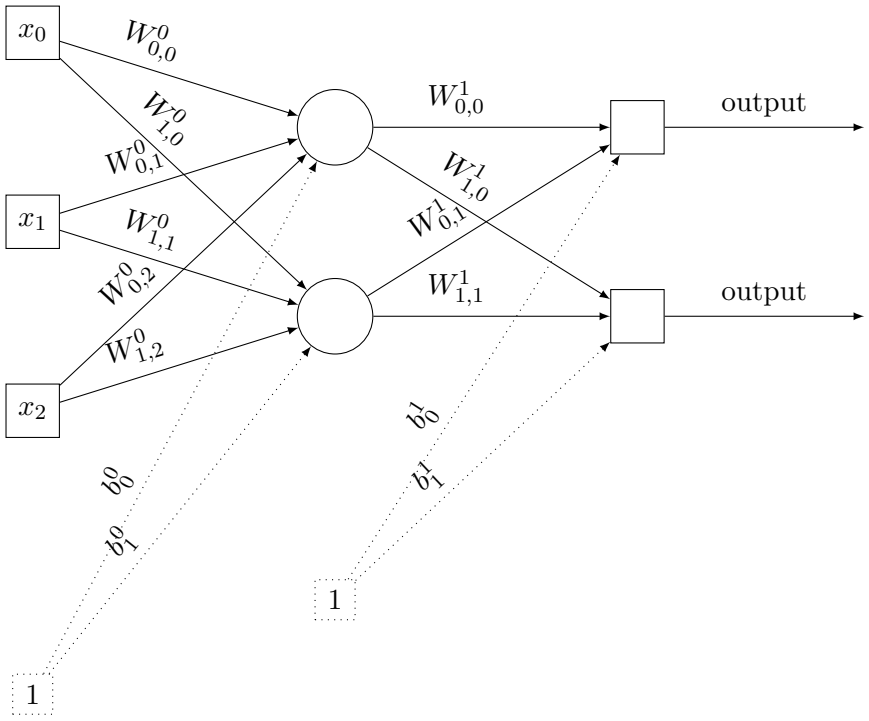


Figure 1.5: An example of a shallow neural network $f : \mathbb{R}^3 \to \mathbb{R}^2$. We call a depth 1 FNN **shallow**. The leftmost layer is the input layer with $N = 3$ inputs. The middle layer, which is the only nonlinear layer in this diagram, is a hidden layer of neurons with $d_0 = 2$ neurons.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

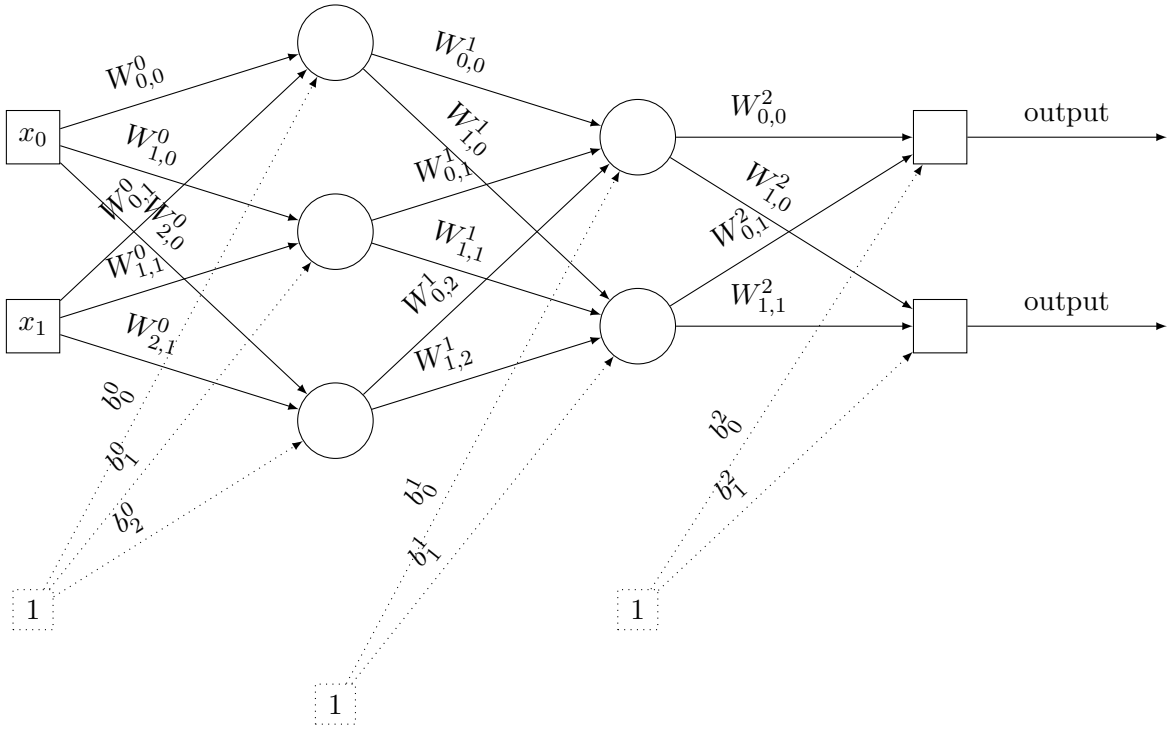


Figure 1.6: An example of a neural network $f : \mathbb{R}^2 \to \mathbb{R}^2$ of depth $L = 2$. The leftmost layer is the input layer with $N = 2$ inputs. The second layer from the left is the first hidden layer of neurons, which has $d_0 = 3$ neurons. The third layer from the left is the second hidden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

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

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 [20]. 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 [20])**.** *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*

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

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, [20] 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.*

Theorem 1.2.10 (Section 24.3 in [20])**.** *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 [20, 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. □

1.3 TO INCLUDE

1.4 Approximate Counting (CMSE 890 Lecture 1)

As way of introduction, in order provide some interesting and relevant examples that are illustrative of the larger course content, we consider three by now quotidian problems in big data settings.

Counting objects is a common challenge in settings involving very large data sets. Memory efficient methods are needed in order to make object counts feasible for routine use on these data. This type of problem and the ensuing discussion will also serve as an introduction to some key ideas for the course. In it we see a deterministic, simple sounding task (counting in the case) which under further study shows the need for fast and memory efficient algorithms that give good approximations to well constructed statistics questions.

A formal statement of the problem is as follows: Given a sequence $\{z_j\}_{j=1}^N$ where $\forall j$, $z_j \in U$ and some item $w \in U$ of interest, count the number of occurrences of term w in the sequence $\{z_j\}_{j=1}^N$.

Goal. *Estimate the count of w occurring in the sequence using $\lceil \log_2 \lceil \log_2 N \rceil \rceil$ bits of memory. We require our estimate of the count be larger than the actual count, but no more than twice the actual count.*

The source of the overestimate error on the count will be made clear shortly. Examples abound for data sets for which counts of this sort are useful

Example 1.4.1. *U is the set of all possible phone numbers, and $\{z_j\}_{j=1}^N$ is a list of phone numbers which have communicated with a particular cellphone tower over some period of time. The term w is a phone number of interest, perhaps a known spammer.*

Example 1.4.2. *U is all possible pairs of words in the English language. So* `hello world` *or* `thank you` *are members of U. The sequence $\{z_j\}_{j=1}^N$ is a list of all pairs of words that appear in emails contained in some user's inbox. The term w then could be* `buy pepsi` *which is of interest to perhaps stock traders or advertisers.*

Example 1.4.3. *U is all possible IP addresses and $\{z_j\}_{j=1}^N$ is a list that contains the originating IP address for all packets received by a certain router. The term w is the IP address of a server used by a movie streaming service of interest to an internet service provider.*

We may wish to consider counts of many different terms w for say all cell-phone towers in a particular country, or all users of some particular email service. Clearly, the size of such data sets means that counts can be potentially very large. Since N is an integer, a priori, we would need (maximally) $\lfloor \log_2 N \rfloor + 1$ bits to store a count of each w.

Note. *We can store N using $\lfloor \log_2 N \rfloor + 1$ bits. We have $\lfloor \log_2 N \rfloor = k$ only if $k \leq \log_2 N < k+1$ if and only if $2^k \leq N < 2^{k+1}$. That is, $2^k \leq N < 2^{k+1}$ is the range of integers which requires $k+1$ bits. So the integers requiring 4-bits for example are 8 through 15. Depending on implementation there are other bits required to say, store the sign of the integer. For simplicity we say that storing an integer of size N requires $\lceil \log_2 N \rceil$ bits, though this may be off by one, or some other constant, depending on implementation.*

A first, naive approach is to increment a counter after one scan of the sequence, and then store the logarithm of that count.

Algorithm 1 Naive Counter

Input: $\{z_j\}_{j=1}^N, w$
Output: approximate count of w in $\{z_j\}_{j=1}^N$
for $j = 1$ to N **do**
  if $z_j = w$ **then**
    $\tilde{w} \leftarrow \tilde{w} + 1$
  end if
end for
$E \leftarrow \lceil \log_2 \tilde{w} \rceil$

Since E is of size at most $\lceil \log_2 N \rceil$ it takes at most $\lceil \log_2 \lceil \log_2 N \rceil \rceil$ bits to store. Due to the information lost by taking the ceiling, we also have that $\tilde{w} \leq 2^E \leq 2\tilde{w}$.

Question 1.4.4. *Does E and the algorithm 15 achieve our goal?*

No. While it is true that E occupies the right number of bits, the counter itself $\tilde{w}$ would need to occupy possibly $\lceil \log_2 N \rceil$ bits when running the algorithm.

Another problem which is similar to counting objects is the distinct elements problem. Here we concerned with determining the number of distinct elements which appear in a given sequence, as opposed to the frequency.

Formally, given a sequence $\{z_j\}_{j=1}^N$ where $\forall j$, $z_j \in U$, and $|U| = D$ we wish to compute the cardinality of $\{z_j\}_{j=1}^N$ as a set.

Goal. *Estimate the number of distinct elements in a sequence using a number of bits independent of both N and D.*

One can imagine many different settings where such a count of distinct elements would be useful.

Example 1.4.5. *U is the set of all possible phone numbers, and $\{z_j\}_{j=1}^N$ is a list of phone numbers which have communicated with a particular cellphone tower over some period of time. The cardinality of the sequence as a set would be the number of unique cellphones that used the tower.*

Example 1.4.6. *U is all possible pairs of words in the English language. The sequence $\{z_j\}_{j=1}^N$ is a list of all pairs of words that appear in a user's current email outbox. The cardinality of the sequence as a set would be an indicator of the variation of a given user's word choice in writing emails.*

We consider two naive solutions to this problem, and observe how they do not achieve the stated goal.

Algorithm 2 Naive Distinct Elements by D-array

Input: $\{z_j\}_{j=1}^N$
Output: number of distinct elements in $\{z_j\}_{j=1}^N$
Let $A :=$ array of zeros of size D
for $j = 1$ to N **do**
  if $A[z_j] = 0$ **then**
    $A[z_j] := 1$
  end if
end for
$\|A\|_0$

Algorithm 3 Naive Distinct Elements by Sorted List

Input: $\{z_j\}_{j=1}^N$
Output: number of distinct elements in $\{z_j\}_{j=1}^N$
Let $L[j] := z_j$
Sort L
for $j = 1$ to $N - 1$ **do**
  if $L[j] = L[j + 1]$ **then**
    flag $L[j + 1]$ for removal
  end if
end for
$|L|$

However, neither of these algorithms meet the requirements of the stated goal. In the case of algorithm 19 the array A clearly occupies D bits. In the case of 20, the list L needs N entries, and so will occupy at least N bits of memory. In a future lecture, we will study the Flajolet-Martin Algorithm which does solve the distinct element problem with constant memory.

The third problem we consider is Nearest Neighbor in $\mathbb{R}$. Here we have a set of points, and are presented with a query point and wish to return the closet point in our set to the query point, reckoned by a norm of interest. Formally, we have $S \subset R^D$, and query $\mathbf{y} \in R^D$

and compute $\mathbf{y}_{NN} = \arg\min \|\mathbf{x} - \mathbf{y}\|$. The set S has cardinality N which can be very large. Naturally we can extend this to k-nearest neighbors by returning a list of the k closest points.

A simple linear scan then of the set is perhaps the most obvious solution to the problem

Algorithm 4 Naive Nearest Neighbors

Input: $S, \mathbf{y}, \|\cdot\|$
Output: $\mathbf{y}_{NN}$
$d = \infty$
for x in S **do**
  if $\|\mathbf{x} - \mathbf{q}\| < d$ **then**
    $\mathbf{y}_{NN} \leftarrow \mathbf{x}$
  end if
end for

This problem is a fundamental building block type of problem in many algorithms and data science applications.

Example 1.4.7. *S is the a database of gray-scale images. A query point q is a novel image, we return the image that is closest to using the ℓ_1 norm*

Example 1.4.8. *S is a database of names of people who bought departing tickets from a given airport. A query point q is a name of a passenger of interest, we return the name that is closest to it using the Hamming distance.*

Example 1.4.9. *S is a database of users of a dating website. Each user has a vector of different features, which is computed from data collected about their interests, hobbies, preferences, etc. A query point q represents a particular user, and developers for the website have engineered a norm which represents similarity between users. The closest point in S is recommended as a potential partner.*

Since each of the N points in S needs to be compared to the query point, and calculating the norm of the difference depends on the dimension D of the space, this scan has $\mathcal{O}(ND)$ complexity. We will later study how to improve on this using good approximations.

Definition 1.4.10. *For $p \geq 1$, the ℓ_p-norm of a vector in $\mathbf{x} \in \mathbb{C}^D$ is a map $\|\cdot\|_p : \mathbb{C}^D \to [0, \infty)$ defined by*

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^{D} |x_j|^p \right)^{1/p}$$

if $p = \infty$ then $\|\mathbf{x}\|_\infty = \max_j |x_j|$

Exercise 1.4.1. *Given norms* $\|\cdot\|_\dagger$ *and* $\|\cdot\|_\star$ *on* $\mathbb{C}^D$ *and* $\alpha, \beta \in [0, \infty)$*, prove that* $\|\mathbf{x}\|_+ = \alpha\|\mathbf{x}\|_\dagger + \beta\|\mathbf{x}\|_\star$ *is also a norm in* $\mathbb{C}^D$

Other applications that will be relevant to our study in this course are

1. Fast Monte Carlo integration approximation
2. Fast approximate solutions to classic numerical linear algebra problems in the big data setting such as
   (a) least square regression
   (b) matrix-matrix multiplication
   (c) Principal Component Analysis
3. Compressive sensing
4. Heavy Hitter problems

Heavy Hitter problems refer to cases where we want to find those values which occur most frequently in a large (streaming) sequence of data. For example, a seller such as Walmart may be interested in the hundred most purchased items across many different stores on a minute by minute or second by second time-frame. Another subtype of Heavy Hitter problem appears in group testing. Here, many specimens are collected and tested together in batches. So for example, 20 patients may submit specimens that are combined into batches containing samples from 5 different specimens, and say samples from each specimen are included in 3 different batches which are then tested for the disease. If the prevalence of the disease is sufficiently small, batching schemes can be designed to economize testing but still ensure identifiability of patients who have the disease.

1.5 Fast Function Approximation via Compressive Sensing (MTH 994 Lecture 1)

The main problem that the course addresses is as follows

Design an algorithm, i.e. a computable function, $\Delta : \mathbb{C}^m \to \mathbb{C}^N$ where $m \leq N$ and a set of linear measurements $\Phi \in \mathbb{C}^{m\times N}$ for a given subset $\mathcal{F}_{\mathbf{p}} \subset \mathbb{C}^N$ with parameters $\mathbf{p} \in \mathbb{C}^r$ such that for all $(\mathbf{n}, \mathbf{x}) \in \boldsymbol{Z} \times \mathcal{F}_{\mathbf{p}}$ the following holds

$$\|\Delta(\Phi\mathbf{x} + \mathbf{n}) - \mathbf{x}\|_X \leq C_{\mathbf{p},X,Y} \inf_{\mathbf{y}\in F_{\mathbf{p}}} \|\mathbf{x} - \mathbf{y}\|_Y + \tilde{C}_{\mathbf{p},X,Y,Z}\|\mathbf{n}\|_Z + \epsilon_{\mathbf{p},X,Y,Z} \tag{1.5}$$

In effect, what we seek is a reconstruction or invertibility property for our algorithm, namely, $\Delta(\Phi\mathbf{x}) = \mathbf{x}$. We know that this property cannot hold in the generic case where $\mathbf{x} \in \mathbb{C}^N$ since $m \leq N$ and thus the null space of Φ will be at least of dimension $N - m$.

So, the condition that $\mathbf{x} \in \mathcal{F}_{\mathbf{p}}$ makes the desired property possible, and the nature of $\mathcal{F}_{\mathbf{p}}$ crucial to our understanding and solution to the problem.

Some key remarks for equation 1.5:

1. The algorithm $\Delta : \mathbb{C}^m \to \mathbb{C}^N$ should be implementable in a manner that is fast, memory efficient, and robust to noise.
2. The norms $\|\cdot\|_{X,Y,Z}$ will usually be ℓ_p-norms for $p = 1, 2, \ldots$
3. $\mathbf{n} \in \mathbb{C}^m$ is arbitrary noise on the input $\Phi\mathbf{x}$. Deterministic or probabilistic pertubations to the input are both possible and the more general the case we can accommodate in our algorithm the better.
4. $\epsilon_{\mathbf{p},X,Y,Z} \in \mathbb{R}^+$ is a small error. This can be round-off error, though often in the sequel we will take it to be zero.
5. Constants like $C_{\mathbf{p},X,Y}$ are absolute in the sense that they are independent of any particular $\mathbf{x}$ and noise $\mathbf{n}$.
6. Often, we consider the compressed measurement case, where for $\Phi \in \mathbb{C}^{m\times N}$ we have $m \ll N$.
7. $\mathcal{F}_{\mathbf{p}} \subset \mathbb{C}^N$ will be some geometrically simple set parameterized by $\mathbf{p}$, such as
   (a) (Manifold) $\mathcal{M}_{[d,\tau]}$, a d-dimensional sub-manifold of $\mathbb{R}^N$ whose reach is bounded by τ. There are other possible parameters, such as volume or diameter which could be used to describe the geometry of the manifold.
   (b) (Compressed sensing) $K_s \subset \mathbb{C}^N$, where K_s is the set of s-sparse vectors in $\mathbb{C}^N$, i.e $\left\{\mathbf{x} \in \mathbb{C}^N | \|\mathbf{x}\|_0 \leq s\right\}$ which is equivalently $\bigcup_{S\subset[N],|S|=s} \operatorname{span}\{\mathbf{e}_j\}_{j\in S}$ where $\mathbf{e}_j$ are the standard basis vectors. That is, the span of vectors with s non-zero entries.

Note that when $\epsilon_{\mathbf{p},X,Y,Z} = 0$ and in the absence of noise, $\mathbf{n} = \mathbf{0}$, equation 1.5 implies the invertibility property, $\Delta(\Phi\mathbf{x}) = \mathbf{x}$, $\forall \mathbf{x} \in \mathcal{F}_{\mathbf{p}}$, which is equivalent to the following, by the linearity of Φ

$$\mathbf{x} \neq \mathbf{y} \iff \Phi(\mathbf{x} - \mathbf{y}) \neq 0 \, \forall \mathbf{x}, \mathbf{y} \in \mathcal{F}_{\mathbf{p}} \tag{1.6}$$

However, numerically 1.6 is not a tenable, realistic property to design around. So we define a stronger property which will imply 1.6.

This is known as the Johnson-Lindenstrauss (JL) embedding property. We say that Φ has the JL-property when $\exists \epsilon \in (0, 1)$ such that

$$|\|\Phi(\mathbf{x} - \mathbf{y})\|_X^2 - \|\mathbf{x} - \mathbf{y}\|_Y^2| \leq \epsilon \|\mathbf{x} - \mathbf{y}\|_Y^2, \, \forall \mathbf{x}, \mathbf{y} \in \mathcal{F}_{\mathbf{p}} \tag{1.7}$$

Analyzing this property in terms of different spaces and matrices will occupy much of our subsequent study. We now conclude the introduction lecture with a discussion of function approximation, and how it relates to the key property 1.7 larger goals of the course.

Running Example. *Suppose $\mathcal{D} = [0,1]^D$, the D-dimensional cube, and $f \in \mathcal{H}$ for $\mathcal{H} = L^2_\mu(\mathcal{D}, \mathbb{C})$, the separable Hilbert space of square intergrable complex valued functions on domain $\mathcal{D}$*

1. *Pick a countable orthornormal basis $\mathcal{B}$ of $\mathcal{H}$.*

$$\mathcal{B} = \{b_j\}_{j \in \mathbb{Z}^D}$$

For example, $\mathcal{B}$ is the Fourier basis. Note that through a Gram-Schmidt process we are guaranteed the existence of a maximal orthonormal set in seperable Hilbert space.

2. *Pick a finite subset $\mathcal{B}' \subset \mathcal{B}$ with $|\mathcal{B}'| = N$. For example, $\mathcal{B}'$ corresponds to some frequencies such as those in a hyperbolic cross, or frequencies in $(\mathbb{Z} \cap [-M, M])^D$ for some $M \in [0, \infty)$*

3. *Approximate f by its projection $P_{\mathcal{B}'} f = \sum_{j \in \mathcal{I}} b_j \langle b_j, f \rangle$ where $\mathcal{I}$ is the index set corresponding to the finite basis, $\mathcal{I} = \left\{ j \in \mathbb{Z}^D | b_j \in \mathcal{B}' \right\}$, so $|\mathcal{I}| = N = |\mathcal{B}'|$*

Given m input measurements $\langle a_1, f \rangle, \ldots, \langle a_m, f \rangle$ we can restate the example in terms of 1.7 by setting $\mathbf{x} \in \mathbb{C}^N$ to $x_j = \langle b_j, f \rangle$ and $\Phi \in \mathbb{C}^{m \times N}$, $\Phi_{\ell,j} = \langle a_\ell, b_j \rangle$, $\forall j \in \mathcal{I}$ such that each input measurement satisfies

$$\begin{aligned}
\langle a_\ell, f \rangle &= \langle a_\ell, P_{\mathcal{B}'} f \rangle + \underbrace{\langle a_\ell, (I - P_{\mathcal{B}'}) f \rangle}_{n_\ell} \\
&= \langle a_\ell, \sum_{j \in \mathcal{I}} b_j \langle b_j, f \rangle \rangle + n_\ell \\
&= \sum_{j \in \mathcal{I}} \langle a_\ell, b_j \rangle x_j + n_\ell \\
&= \sum_{j \in \mathcal{I}} \Phi_{\ell,j} x_j + n_\ell
\end{aligned}$$

Note the noise vector $\mathbf{n}$ is due to the truncation error incurred by using only a finite number of basis elements in this function approximation setting. So, if we also have that $\mathbf{x}$ is in (or near) $\mathcal{F}_p \in \mathbb{C}^N$ then the conclusion of the running example is indeed statement of the result 1.7.

Note that a large number of basis elements may be required to reduce error of the approximation, especially for high dimensional input space D. This means that computationally, function approximation may be intractable unless we use the structure of $\mathcal{F}_\mathbf{p}$ to compress the computation of $\sum_{j \in \mathcal{I}} \Phi_{\ell,j} x_j$.

Example 1.5.1 (Function Approximation, 1-D sparse Fourier Transform)**.** *Suppose* $f : [0,1] \to \mathbb{C}$, $f \in L^2([0,1],\mathbb{C}) \cap C^1([0,1])$. *Choose the orthonormal basis* $\mathcal{B} = \left\{e^{2\pi kix}\right\}_{k\in\mathbb{Z}}$ *and select the finite subset* $\mathcal{B}' = \left\{e^{2\pi kix}\right\}_{k\in[-N,N]\cap\mathbb{Z}}$.

Let the input measurements be point samples inside the domain, i.e. $a_\ell = \delta_{x_\ell}$ *where* $\delta_{x_\ell} = \delta(x - x_\ell)$ *for* $x_\ell \in [0,1]$, $\forall \ell \in [m]$. *Choose* $\mathcal{F}_{\mathbf{p}}$ *to be* $\mathcal{K}_s$, *the* s*-sparse vectors in the Fourier basis,* $x = \hat{f}\big|_{k\in[-N,N]\cap\mathbb{Z}}$ *has at most* s *non-zero entries (alternatively we may relax this and say that the bulk of the energy of* x *is in at most* s*-entries).*

Note that Φ *has several constraints - it's entries are now taken from point evaluations of different basis functions at the different sample points* x_ℓ*; and yet we still require that it preserves the norms of vectors in* $\mathcal{F}_{\mathbf{p}}$ *as stated in 1.7. Additionally, in order to achieve an improvement in the speed of our algorithm, we need to improve on the usual Fast Fourier Transform sampling complexity. Instead of the bound* $m \leq N \log N$ *we want* $m \leq s \log^C N$ *where* C *is a small positive, absolute constant. In this way we can benefit from the sparsity of* $\mathcal{F}_{\mathbf{p}}$*. Lastly, we want to be able to recover* x*, i.e. find* $\Delta : \mathbb{C}^m \to \mathbb{C}^N$ *that is able to run in* $\mathcal{O}(s \log^C N)$ *(in contrast to* $\mathcal{O}(N \log N)$ *required for the standard* $FFT^{-1}(FFT(x))$*)*

Exercise 1.5.1. *Prove that 1.7 implies 1.6*

Exercise 1.5.2. *Prove that 1.7 implies*

$$|\|\Phi(\mathbf{x}-\mathbf{y})\|_X - \|\mathbf{x}-\mathbf{y}\|_Y| \leq \epsilon \frac{\|\mathbf{x}-\mathbf{y}\|_Y}{1+\sqrt{1-\epsilon}} \leq \frac{\epsilon\|\mathbf{x}-\mathbf{y}\|_Y}{2-\epsilon}$$

1.6 Tensor Applications

Definition 1.6.1. *An* n*-mode or order-*n *tensor (or* n*-th order) is an* n *dimensional array of complex values, written as*

$$\mathcal{A} \in \mathbb{C}^{I_0 \times I_1 \times \cdots \times I_{n-1}}$$

where $I_j \in \mathbb{N}, j \in [n]$. *An* n*-mode tensor's entries are indexed by a vector* $\mathbf{i} \in [I_0] \times [I_1] \times \cdots \times [I_{n-1}]$ *where* $(\mathcal{A})_{\mathbf{i}} = a_{\mathbf{i}} = a_{i_0,\ldots,i_{n-1}} \in \mathbb{C}$

Example 1.6.2 (1 and 2 mode tensors)**.**

1. *A 1-mode tensor,* $\mathbf{a} \in \mathbb{C}^{I_0}$ *is a vector with entries* $a_j \in \mathbb{C}$, $j \in [I_0]$. *We will denote 1-mode tensors, vectors, the usual way with bolded lowercase letters*

2. *A 2-mode tensor,* $A \in \mathbb{C}^{I_0 \times I_1}$ *is a matrix with entries* $a_{i_0,i_1} \in C$ *for* $i_0 \in [I_0], i_1 \in [I_1]$. *Equivalently* $a_{\mathbf{k}}$, $\mathbf{k} \in [I_0] \times [I_1]$. *We will denote 2-mode tensors, matrices, the usual way with capital un-bolded letters.*

We introduce some terminology that will be useful when describing tensors

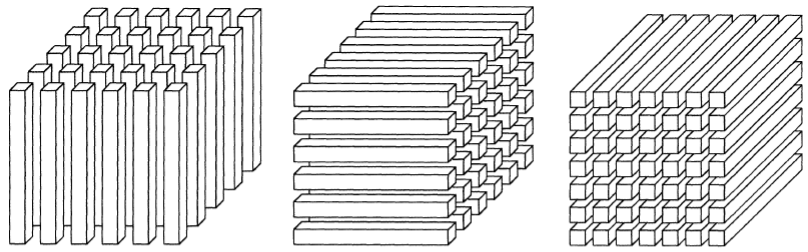
Definition 1.6.3 (Fiber). *Fibers are 1-dimensional subsets of an n-mode tensor. They are formed by fixing $n-1$ of the dimensions and then ranging over all indices in the remaining dimension. So for any $k \in [n]$, and $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{n-1}}$ then a k-mode fiber would be a vector $\mathbf{a} \in \mathbb{C}^{I_k}$ where indices $i_0, \ldots, i_{k-1}, i_{k+1}, \ldots, i_{n-1}$ are fixed, i.e. using* `Matlab` *notation*

$$(\mathcal{A})_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{n-1}} = \mathbf{a}_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{n-1}}$$

So for example, given a matrix $A \in \mathbb{C}^{I_0 \times I_1}$ then $A_{i,:} = \mathbf{a}_{i,:} \in \mathbb{C}^{I_1}$ is a mode-2 fiber (i.e. row). A mode-1 fiber, $\mathbf{a}_{:,j}$ is a column of the matrix.

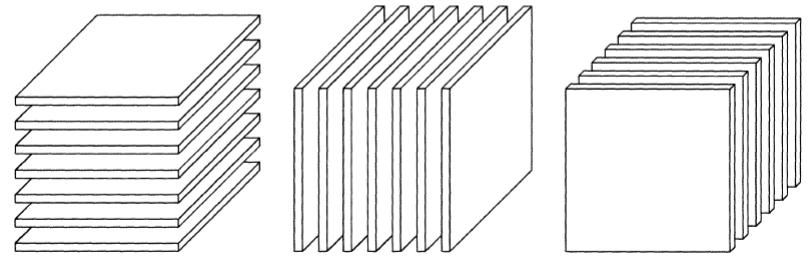
Definition 1.6.4 (Slice). *A matrix formed by varing 2 indices and fixing all other indices of a tensor. That is, suppose $j, k \in [n]$ where $j \neq k$ then*

$$A = \mathcal{A}_{i_0,\ldots,i_{j-1},:,i_{j+1},\ldots,i_{k-1},:,i_{k+1},\ldots,i_{n-1}} \in \mathbb{C}^{I_j \times I_k}$$



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Fig. 2.1 *Fibers of a 3rd-order tensor.*



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Fig. 2.2 *Slices of a 3rd-order tensor.*

Figure 1.8: Figure seen in [35]

Definition 1.6.5 (Sub-tensor). *A k-subtensor of an n-mode tensor ($k < n$) is denoted by a vector of length $n-k$ of indices and a set of k mode indices from the set $[n]$. That is given distinct $j_0, \ldots, j_{k-1} \in [n]$ and define vector $\mathbf{i} \in \bigotimes_{i \neq j_\ell}[I_i]$ of length $n-k$. Let*

$$\mathcal{A}_{j_0,\ldots,j_{k-1},\mathbf{i}} \in \mathbb{C}^{I_{j_0} \times \cdots \times I_{j_{k-1}}}$$

Using this subtensor notation then a mode-k fiber is a subtensor $\mathcal{A}_{k,\mathbf{i}}$ where $k \in [n]$ and $\mathbf{i} \in [I_0] \times \ldots [I_{k-1}] \times [I_{k+1}] \times \cdots \times [I_{n-1}]$. There are $\prod_{\ell \neq k} I_\ell$ potentially different mode-k fibers, one for each possible $\mathbf{i}$.

A slice then is denoted $\mathcal{A}_{\ell,k,\mathbf{i}} \in \mathbb{C}^{I_\ell \times I_k}$. There are $\prod_{j \neq \ell,k} I_j$ slices of dimension $I_\ell \times I_k$.

Next we will discuss reshaping operators - this involves many different possible ways of changing the dimensions of tensors so that they have the same number of entries.

Definition 1.6.6 (Vectorization). *For $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{n-1}}$ $vec(\mathcal{A}) = \mathbf{a}$ where $\mathbf{a} \in \mathbb{C}^{\prod_{k=0}^{n-1} I_k}$*

Definition 1.6.7 (Mode-k Flattening). *For $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{n-1}}$ the k-mode flattening is a matrix $A^{(k)} \in \mathbb{C}^{I_k \times \prod_{j \neq k} I_j}$. We have effectively made the k-th dimension into the rows of the matrix, and the columns are then the different mode-k fibers. In particular $\left(A^{(k)}\right)_{j,\ell} = \mathcal{A}_{\ell_1,\ldots,\ell_{k-1},j,\ell_{k+1},\ldots,\ell_{n-1}}$. The columns are the fibers $\mathcal{A}_{k,\mathbf{i}}$.*

Definition 1.6.8 (Reshaping). *We can reshape an n-mode into any other m-mode tensor with a reshaping operation $R : \mathbb{C}^{I_0 \times \cdots \times I_{n-1}} \to \mathbb{C}^{J_0 \times \cdots \times J_{m-1}}$ provided*

$$\prod_{j=0}^{n-1} I_j = \prod_{\ell=0}^{m-1} \tilde{J}_\ell$$

k-mode flattening and vectorization are two particular reshaping operations.

What is the underlying vector space we can use to study tensors? To answer that, we consider the following norm, inner-product, and operations on tensors:

Definition 1.6.9 (2-norm of a Tensor). *For $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{n-1}}$ then given any $k \in [n]$ we have*

$$\|\mathcal{A}\|_2^2 = \|A^{(k)}\|_2^2 = \|\mathbf{a}\|_2^2 = \sum_{\mathbf{i} \in I} |a_{\mathbf{i}}|^2$$

where $I = [I_0] \times \cdots \times [I_{n-1}]$

Definition 1.6.10 (Inner-product). *For tensors $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{I_0 \times \cdots \times I_{n-1}}$ then*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{\mathbf{i} \in I} a_{\mathbf{i}} \bar{b}_{\mathbf{i}} = \langle \mathbf{a}, \mathbf{b} \rangle$$

that is, the inner-product of the vectorization of the tensors. Note that this is equivalent to $\langle A^{(k)}, B^{(k)} \rangle_{HS} = Trace(A^k (B^{(k)})^*)$, $\forall k \in [n]$, the Hilbert-Schmidt inner product for matrices

Addition and scalar multiplication work component-wise, i.e.

$$(\mathcal{A} + \mathcal{B})_{\mathbf{i}} = a_{\mathbf{i}} + b_{\mathbf{i}}$$

$$(\alpha \mathcal{A})_{\mathbf{i}} = \alpha a_{\mathbf{i}}, \, \forall \alpha \in \mathbb{C}$$

1.6.1 Restricted Inner and Matrix Products

Given long vectors, as may result from reshaping a tensors for example, we may perform inner products with smaller vectors by using well chosen samples or parts from the longer vectors.

Definition 1.6.11. *k-mode product of $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$ and $U \in \mathbb{C}^{J_k \times I_k}$ for $k \in [d]$ is a tensor in $\mathbb{C}^{I_0 \times \cdots \times I_{k-1} \times J_k \times I_{k+1} \times \cdots \times I_{d-1}}$ denoted by*

$$(\mathcal{A} \times_k U)_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{d-1}} = U\mathcal{A}_{i_0,\ldots,\ldots,i_{d-1}}$$

In other words, the k-mode product applies the matrix U to all the mode-k fibers of the tensor $\mathcal{A}$. For example, suppose $\mathcal{A} \in \mathbb{C}^{5\times3\times2}$ and $U \in \mathbb{C}^{4\times5}$. Then $\mathcal{A} \times_1 U \in \mathbb{C}^{4\times3\times2}$ where each of the mode-1 fibers is now the product of $U\mathcal{A}_{:,j,\ell}$, for some $j \in [3], \ell \in [2]$.

In the 2-mode tensor case, i.e., matrices, the usual matrix-matrix multiplication can be understood in terms of 1-mode tensor product.

$$\begin{aligned} AB &= A \left[\begin{array}{c|c|c|c} \mathbf{b}_1 & \mathbf{b}_2 & \ldots & \mathbf{b}_n \end{array} \right] \\ &= \left[\begin{array}{c|c|c|c} A\mathbf{b}_1 & A\mathbf{b}_2 & \ldots & A\mathbf{b}_n \end{array} \right] \\ &= B \times_1 A \end{aligned}$$

Lemma 1.6.12. $(\mathcal{A} + \mathcal{B}) \times_k U = \mathcal{A} \times_k U + \mathcal{B} \times_k U$

Proof. For any $i_0 \in I_0, \ldots, i_{k-1} \in I_{k-1}, i_{k+1} \in I_{k+1}, \ldots, i_{d-1} \in I_{d-1}$ we have

$$\begin{aligned} [(\mathcal{A} + \mathcal{B}) \times_k U]_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{d-1}} &= U(\mathcal{A} + \mathcal{B})_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{d-1}} \\ &= U\mathcal{A}_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{d-1}} + U\mathcal{B}_{i_0,\ldots,i_{k-1},:,i_{k+1},\ldots,i_{d-1}} \\ &= \mathcal{A} \times_k U + \mathcal{B} \times_k U \end{aligned}$$

□

Lemma 1.6.13 (Properties of k-mode products)**.** *Let $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{I_0,\ldots,I_{d-1}}$, $\alpha, \beta \in \mathbb{C}$, $U_\ell, V_\ell \in \mathbb{C}^{m_\ell \times I_\ell}$, $\forall \ell \in [d]$. Then*

1. $(\alpha\mathcal{A} + \beta\mathcal{B}) \times_j U_j = \alpha(A \times_j U_j) + \beta(\mathcal{B} \times_j U_j)$
2. $\mathcal{A} \times_j (\alpha U_j + \beta V_j) = \alpha(\mathcal{A} \times_j U_j) + \beta(\mathcal{A} \times_j V_j)$ *that is, k-mode product is bilinear*
3. *If $j \neq \ell$ then*
$$(\mathcal{A} \times_j U_j) \times_\ell V_\ell = (\mathcal{A} \times_\ell V_\ell) \times_j U_j$$
Note that the run-time complexity is the same regardless of the order one applies the k- or j-mode products
4. *If $W \in \mathbb{C}^{p\times m_j}$ then $(\mathcal{A} \times_j U_j) \times_j W = \mathcal{A} \times_j (WU_j) \in \mathbb{C}^{I_0 \times I_{j-1} \times p \times I_{j+1} \times \ldots I_{d-1}}$*

Definition 1.6.14 (Kronecker Product)**.** *The Kronecker product of two matrices $U \times \mathbb{C}^{m\times n}$ and $V \in \mathbb{C}^{p\times q}$ is a matrix*

$$U \otimes V = \begin{pmatrix} u_{11}V & \dots & u_{1n}V \\ \vdots & & \vdots \\ u_{m1}V & \dots & u_{mn}V \end{pmatrix}$$

where $U \otimes V \in \mathbb{C}^{mp\times nq}$

Lemma 1.6.15. *Let $\mathcal{A} \in \mathbb{C}^{I_0,\dots,I_{d-1}}$, $U_\ell \in \mathbb{C}^{m_\ell \times I_\ell}$ then*

1. $(\mathcal{A} \times_j U_j)^{(j)} = U_j \mathcal{A}^{(j)} \in \mathbb{C}^{m_j \times \prod_{\ell\neq j} I_\ell}$
2. $(\mathcal{A} \times_0 U_0 \times_1 U_1 \cdots \times_{d-1} U_{d-1})^{(j)} = U_j \mathcal{A}^{(j)} \left(U_{d-1} \otimes U_{d-2} \otimes \cdots \otimes U_{j+1} \otimes U_{j-1} \otimes \cdots \otimes U_0\right)^T$

we have assumed a column-major convention for matricization.

In order for the identity seen in Lemma 1.6.15 to hold, we need specify our precise matricization convention.

In our convention, the entry at location $(i_0, \dots, i_{d-1})$ in $\mathcal{A} \in \mathbb{C}^{I_0\times\cdots\times I_{d-1}}$ is located in the matrix as entry (i_n, j) where

$$j = \sum_{\substack{k=0 \\ k\neq n}}^{d-1} i_k J_k$$

where

$$J_k = \prod_{\substack{m=0 \\ m\neq n}}^{k-1} I_m$$

set $J_k = 1$ if the index of the product above is empty.

Example 1.6.16 (3-mode)**.** *The following example appears in [35]: Consider a tensor $\mathcal{A} \in \mathbb{R}^{3\times4\times2}$, the frontal slices are as follows*

$$\mathcal{A}_{:,:,0} = \mathcal{A}_0 = \begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}, \mathcal{A}_{:,:,1} = \mathcal{A}_1 = \begin{pmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{pmatrix}$$

The three different unfoldings are then as follows. Consider $n = 0$,

$$\mathcal{A}^{(0)} = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{pmatrix}$$

Note

$$J_0 = \prod_{\substack{m=0 \\ m\neq 0}}^{0-1} I_m = 1, J_1 = \prod_{\substack{m=0 \\ m\neq 0}}^{1-1} I_m = 1, J_2 \prod_{\substack{m=0 \\ m\neq 0}}^{2-1} I_m = I_1 = 4$$

Now to see how to locate a particular entry, note that $\mathcal{A}_{1,2,1} = 20$*, so in our unfolding* $\mathcal{A}^{(k)}_{(i,j)}$ *we can simply copy the index that corresponds to the* n*-th mode, i.e. the first here,* $i = 1$*. To find the column, compute* $j = \sum_{\substack{k=0 \\ k\neq n}}^{d-1} i_k J_k = 2(1) + 1(4) = 6$*. So our entry with value* 20 *is in location* $(1, 6)$*.*

Consider $n = 1$*,*

$$\mathcal{A}^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{pmatrix}$$

Again, to locate our entry,$\mathcal{A}_{1,2,1} = 20$*, we return the index on the* n*-th mode as our row,* $i = 2$*, and repeat the same calculation for* j*:*

Note

$$J_0 = \prod_{\substack{m=0 \\ m\neq 1}}^{0-1} I_m = 1, J_1 = \prod_{\substack{m=0 \\ m\neq 1}}^{1-1} I_m = I_0 = 3, J_2 \prod_{\substack{m=0 \\ m\neq 1}}^{2-1} I_m = I_0 = 3$$

.

This time we leave out the J_1 *factor:* $j = \sum_{\substack{k=0 \\ k\neq n}}^{d-1} i_k J_k = 1(1) + 1(3) = 4$*. So our entry with value* 20 *is located in* $(2, 4)$*.*

Consider $n = 2$*,* $\mathcal{A}^{(2)}$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & \dots \end{pmatrix}$$

Again, to locate our entry,$\mathcal{A}_{1,2,1} = 20$*, we return the index on the* n*-th mode as our row,* $i = 1$*, and repeat the same calculation for* j*:*

Note

$$J_0 = \prod_{\substack{m=0 \\ m\neq 2}}^{0-1} I_m = 1, J_1 = \prod_{\substack{m=0 \\ m\neq 2}}^{1-1} I_m = I_0 = 3, J_2 \prod_{\substack{m=0 \\ m\neq 2}}^{2-1} I_m = I_0 = 3$$

.

This time we leave out the J_2 *factor:* $j = \sum_{\substack{k=0 \\ k\neq n}}^{d-1} i_k J_k = 1(1) + 2(3) = 7$*. So our entry with value* 20 *is located in* $(1, 7)$*.*

1.6.2 Low Rank Approximation

Our next topic concerns how different methods can be used to approximate tensors. Our first attempt will illustrate the need for better decompositions other than simply reshaping tensors into familiar objects.

Suppose we have q tensors of size $\mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$, $\mathcal{A}_1, \ldots, \mathcal{A}_q$. We will compress the tensors by performing PCA on the vectorized tensors. Our goal then in this case is to solve the minimization problem:

$$\sum_{j=1}^{m} \min_{S_j \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}} \|\mathcal{A}_j - \mathcal{S}_j\|_2^2$$

This is equivalent to

$$\sum_{j=1}^{m} \min_{\mathbf{s}_j \in \mathcal{S} \subseteq \mathbb{C}^{\prod_{m=0}^{d-1} I_m}} \|\mathbf{a}_j - \mathbf{s}_j\|_2$$

where $\text{vec}(\mathcal{A}_j) = \mathbf{a}_j \in \mathbb{C}^{\prod_{m=0}^{d-1} I_m}$. We can use the SVD of the following data matrix

$$\left[\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \ldots & \mathbf{a}_q \end{array} \right] = U \Sigma V^* \in \mathbb{C}^{q \times \prod_{m=0}^{d-1} I_m}$$

Once we have the singular vectors, we can tensorize their outer product using the inverse of our vectorizing reshaping operation. That is

$$\mathcal{A}_j \approx \sum_{k=1}^{m} \sigma_{j,k} \mathcal{T}_k$$

where $\mathcal{T}_k$ are the tensors obtained from the principal directions (obtained from the singular vectors of SVD of the data matrix) and $\sigma_{j,k}$ are the appropriate principal scores (again computed from the singular vectors and singular values of the data matrix).

What compression does this achieve? The space required for our original collection of tensors $\mathcal{A}_1, \ldots, \mathcal{A}_q \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$ is $\mathcal{O}(q \prod_{m=0}^{d-1} I_m)$. After PCA, we need keep m basis tensors of the same dimension $\mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$ and our coordinates or principal scores will also need to be stores and there are mq of these. So the space is $\mathcal{O}(m \prod_{m=0}^{d-1} + qm)$ which is unsatisfactory because the dependence on $\prod_{m=0}^{d-1}$ is unchanged. Additionally, there's no interpretable structure to the basis tensors $\mathcal{T}_k$. This motivates us then to look to another approach for decomposing (and therefore compressing) tensors.

Definition 1.6.17 (Rank one Tensor)**.** *Given d-vectors $\mathbf{x}_j \in \mathbb{C}^{I_j}$ for $j \in [d]$, the outer-product*

$$\mathcal{X} = \bigcirc_{j=0}^{d-1} \mathbf{x}_j \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$$

has entries given the product of corresponding entries of the vectors, i.e.

$$\mathcal{X}_{i_0,\ldots,i_{d-1}} = \left(\bigcirc_{j=0}^{d-1} \mathbf{x}_j\right)_{i_0,\ldots,i_{d-1}} = (\mathbf{x}_0)_{i_0} (\mathbf{x}_1)_{i_1} \ldots (\mathbf{x}_{d-1})_{i_{d-1}}$$

any d-mode tensor where it is possible to write it as such an outer product of d vectors is a rank one tensor.

Note that storing a rank one tensor means storing only the vector components, rather than all entries. This definition in the 2-mode case is the familiar rank one matrix case, for $\mathbf{u} \in \mathbb{C}^m, \mathbf{v} \in \mathbb{C}^N$

$$A = \mathbf{u} \circ \mathbf{v} = \mathbf{u}v^*$$

then matrix $A \in \mathbb{C}^{m \times N}$ is a rank one matrix.

Definition 1.6.18. *$\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$ is a r tensor if it can be written as the sum of r rank one tensors, that is*

$$\mathcal{A} = \sum_{\ell=0}^{r-1} \left(\bigcirc_{j=0}^{d-1} \mathbf{x}_j^{(\ell)}\right)$$

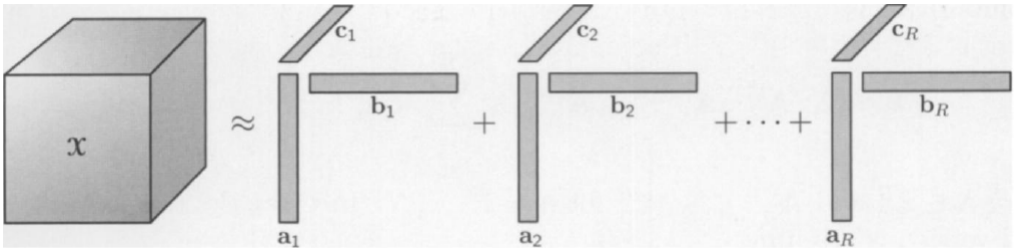


Figure 1.9: Schematically a rank R decomposition for a 3-mode tensor $\mathcal{X}$ as seen in [35]

Note that unlike the PCA example given above, this decomposition does not require us to consider a set of tensors; a single tensor will be decomposable in this fashion. Furthermore, each of the basis tensors in this case does have a simple structure - it can be stored as d vectors and so takes up $\mathcal{O}\left(r\sum_{j=0}^{d-1} I_j\right)$-space.

So with this definition, we ask then how, given a tensor $\mathcal{A}$ can we find its rank r decomposition. How to select or determine r is a question we will set aside for the time being.

Given $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$ we want to find a rank r approximation as

$$\arg \min_{\mathbf{x}_j^{(\ell)} \in \mathbb{C}^{I_j}, j \in [d], \ell \in [r]} \|\mathcal{A} - \sum_{\ell=1}^{r} \left(\bigcirc_{j=0}^{d-1} \mathbf{x}_j^{(\ell)}\right)\|$$

In the generic case, the above optimization problem is difficult. However, the base case of $d = 2$ leads to consider a method which in practice can yield good results, though its gaurauntees are not well understood.

In the event that $d = 2$ then the optimization problem is equivalent to

$$\min_{\alpha_\ell} \|A - \sum_{\ell=0}^{r-1} \alpha_l \mathbf{u}_\ell \mathbf{v}_\ell^*\|_F = \sqrt{\sum_{j=r}^{N-1} \sigma_j(A)}$$

where $\alpha_\ell = \sigma_\ell(A)$ and $\mathbf{u}_\ell, \mathbf{v}_\ell$ are the singular vectors of A. That is, the best rank r approximation to a matrix A is given by the leading r factors from the SVD.

So the idea then for our tensor decomposition is to reduce to the $d = 2$ case. Suppose for the time being that we have $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}$, we know it is rank r and we know all but the first mode vectors in each of the r rank 1 factors. That is, we have

$$\bigotimes_{j>0}^{d-1} \mathbf{x}_j^{(\ell)}$$

for all $\ell \in [r]$. With this (mostly) complete factorization for $\mathcal{A}$, we can find the missing mode by solving a least squares problem.

So,

$$\mathcal{A} = \sum_{\ell=0}^{r-1} \left(\bigotimes_{j=0}^{d-1} \mathbf{x}_j^{(\ell)} \right)$$

Now consider the subtensor $\mathcal{A}_{([d]\setminus\{0\}, i_0)}$. This is the tensor found by fixing an index in the 0-th mode and varying all other indices. Naturally then there are I_0 such subtensors. For any $i_0 \in I_0$ the entries of the subtensor $\mathcal{A}_{([d]\setminus\{0\}, i_0)}$ are equal to

$$\sum_{\ell=0}^{r-1} \left(\mathbf{x}_0^{(\ell)}\right)_{i_0} \bigotimes_{j>0}^{d-1} \mathbf{x}_j^{(\ell)}$$

That is, we have a sum of products of scalar unknowns with $d-2$ outer product - and so after a careful rearrangement of elements, we will have a linear system:

$$\begin{bmatrix} | & | & & | \\ \operatorname{vec}\left(\bigotimes_{j=0}^{d-1} \mathbf{x}_j^{(0)}\right) & \operatorname{vec}\left(\bigotimes_{j=0}^{d-1} \mathbf{x}_j^{(1)}\right) & \cdots & \operatorname{vec}\left(\bigotimes_{j=0}^{d-1} \mathbf{x}_j^{(r-1)}\right) \\ | & | & & | \end{bmatrix} \begin{bmatrix} \left(\mathbf{x}_0^{(0)}\right)_{i_0} \\ \left(\mathbf{x}_0^{(1)}\right)_{i_0} \\ \vdots \\ \left(\mathbf{x}_0^{(0)}\right)_{i_0} \end{bmatrix} = \cdots$$

$$\cdots \begin{bmatrix} | \\ \operatorname{vec}\left(\mathcal{A}_{([d]\setminus\{0\}, i_0)}\right) \\ | \end{bmatrix}$$

Let us denote B_0 as the $\prod_{j=1}^{d-1} I_j \times r$ matrix formed by using the vectorized $d-1$-mode outer products as columns. Note that $\mathcal{A}_{([d]\setminus\{0\},i_0)} = \mathcal{A}^{(k)}_{i_0,:}$, that is the vectorized subtensor is equal to the i_0-th row of the 0-mode unfolding of $\mathcal{A}$

So, in order to solve the missing unknown, we have I_0 overdetermined linear systems of the form $A^{(0)}_{:,i_0} = B_0\mathbf{x}$ to solve in order find all the unknowns. i.e. denoting $\mathbf{x}_\ell = \left(\mathbf{x}^{(\ell)_0}\right)_{i_0}$ for $\ell \in [r]$

$$\mathbf{x} = \arg\min_{\mathbf{y}\in\mathbb{C}^r} \|\mathbf{b} - B_0\mathbf{y}\|_2$$

Solving this for all $i_0 \in [I_0]$.

This can be formulated equivalently as follows

$$\left(\mathcal{A}^{(k)}\right)^T = B_k \left[\begin{array}{c|c|c|c} \mathbf{x}_k^{(0)} & \mathbf{x}_k^{(1)} & \dots & \mathbf{x}_k^{(r-1)} \end{array}\right]^T$$

where we have combined the I_k different vector least square fitting problems into one matrix least square fitting problem of the form

$$\arg\min_{X\in\mathbb{C}^{r\times I_k}} \left\| \left(\mathcal{A}^{(k)}\right)^T - B_k X \right\|_F^2$$

that is X will solve for all the k-mode missing factor vectors.

With this in hand, we are now ready to address the question of how to obtain the complete factorization of an arbitrary rank r tensor – our proceeding formulation only addressed how to find the k-mode missing factor vectors supposing all the other $r(d-1)$ factor vectors were known.

Algorithm 5 Alternating Least Squares Minimization

Input: $\mathcal{A} \in \mathbb{C}^{I_0\times\cdots\times I_{d-1}}$
Output: $\left\{\mathbf{x}_j^{(\ell)}\right\}_{j\in[d],\ell\in[d]}$
Initialize $\left\{\mathbf{x}_j^{(\ell)}\right\}_{j\in[d],\ell\in[r]}$ randomly
for $i = 1$ to maximum iterations **do**
  for $k = 0$ to $d-1$ **do**
    $\left[\begin{array}{c|c|c|c} \mathbf{x}_k^{(0)} & \mathbf{x}_k^{(1)} & \dots & \mathbf{x}_k^{(r-1)} \end{array}\right] \leftarrow \arg\min_{X\in\mathbb{C}^{r\times I_k}} \left\| \left(\mathcal{A}^{(k)}\right)^T - B_k X \right\|_F^2$
  end for
end for
return $\left\{\mathbf{x}_j^{(\ell)}\right\}_{j\in[d],\ell\in[d]}$

Note that the above algorithm requires the solution of (d)`max_iterations` overdetermined least square problems - a potential bottleneck which can be mitigated by using fast approximate least square methods like the one described in Theorem 4.3.3.

Also note that the algorithm is a greedy algorithm an - its convergence properties are not well understood nor does it guarantee any type of global optimality.

Next we turn to another important Tensor decomposition method.

1.6.3 Tucker Rank and Decomposition

Definition 1.6.19. *A tensor $\mathcal{A} \in \mathbb{C}^{I_0 \times \dots I_{d-1}}$ has $(r_1, \dots, r_{d-1})$-Tucker rank if there exists a core tensor $\mathcal{C} \in \mathbb{C}^{r_0 \times \dots r_{d-1}}$ and matrices $U_j \in \mathbb{C}^{I_j \times r_j}$, $\forall j \in [d]$ such that*

$$\mathcal{A} = \mathcal{C} \bigtimes_{j=0}^{d-1} U_j$$

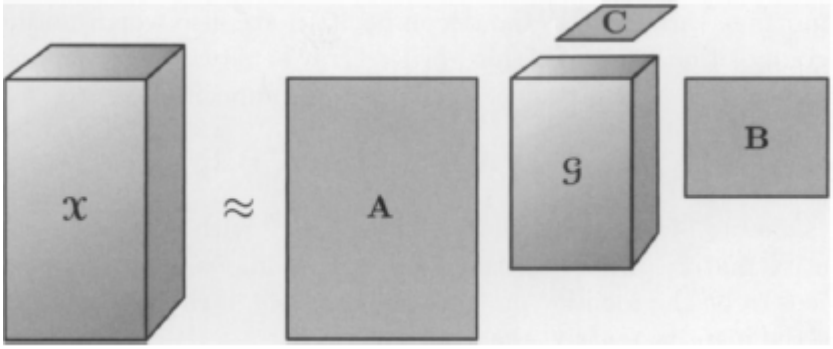


Figure 1.10: Schematically a Tucker decomposition for a 3-mode tensor $\mathcal{X}$ with core tensor $\mathcal{G}$ and factor matrices A, B, C as seen in [35]

Note the space requirement to store a Tucker decomposition of a tensor is $\mathcal{O}\left(\prod_{j=0}^{d-1} r_j + \sum_{j=0}^{d-1} I_j r_j\right)$, where the first term accounts for all the entries of the core tensor and the second term accounts for all entries of the factor matrices. Recall that $\mathcal{O}\left(\prod_{j=0}^{d-1} I_j\right)$ space is required to store the unfactored tensor, and so in the event that the Tucker rank is appreciably smaller than the original mode for at least some of the modes, the Tucker decomposition will occupy significantly less space.

Note that as a convention, U_j can be taken to have orthonormal columns - by orthonormalizing U_j, we can suitably alter $\mathcal{C}$.

To approximate a tensor with a low Tucker rank representation, we can solve the following optimiziation problem

$$\underset{\substack{\mathcal{C} \in \mathbb{C}^{r_0 \times \dots \times r_{d-1}} \\ \{U_j\}_{j \in [d]}, U_j^* U_j = I}}{\arg\inf} \quad \|\mathcal{A} - \mathcal{C} \bigtimes_{j=0}^{d-1} U_j\|_2^2$$

when the number of modes is larger than 2, this optimization problem is difficult to solve. One approach is to use the SVD of each of the d different unfoldings of $\mathcal{A}$

Algorithm 6 Higher Order SVD

Input: $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}, (r_0, \ldots, r_{d-1})$
Output: $\mathcal{C}, \{U_j\}_{j \in [d]}$
Compute r_j-truncated SVD of mode-j unfolding of $\mathcal{A}$

$$A^{(j)} = U_j \Sigma_j V_j^*, \forall j \in [d]$$

$\mathcal{C} \leftarrow \mathcal{A} \times_{j=0}^{d-1} U_j^* \in \mathbb{C}^{r_0 \times \cdots \times r_{d-1}}$
return $\mathcal{C}, \{U_j\}_{j \in [d]}$

Note that for each unfolding, the full SVD has form

$$A^{(j)} = \underbrace{U}_{I_j \times I_j} \underbrace{\Sigma}_{I_j \times \prod_{j \neq k}} I_k \underbrace{V^*}_{I_k \times \prod_{j \neq k} \prod_{j \neq k} I_k}$$

the r_j-truncated SVD has form

$$A^{(j)} \approx \underbrace{U}_{I_j \times r_j} \underbrace{\Sigma}_{r_j \times r_j} I_k \underbrace{V^*}_{r_j \times \prod_{j \neq k} I_k}$$

This problem then is repeated for each of the d different modes. This is potentially a bottleneck computationally and so can likely benefit from fast approximations to the SVD as described in algorithm **??**.

Now, to further improve the decomposition, we can take an approach like 5 and alternate, successively solving for U_j and iterate on this processes.

Algorithm 7 Higher Order Orthogonal Iteration

Input: $\mathcal{A} \in \mathbb{C}^{I_0 \times \cdots \times I_{d-1}}, (r_0, \ldots, r_{d-1})$

Output: $\mathcal{C}, \{U_j\}_{j \in [d]}$

Initalize $\left\{U_j^{(0)}\right\} \leftarrow HOSVD(\mathcal{A})$

for $i = 1$ to M **do**

$\forall j$ update $U_j^{(i-1)}$ by computing

$$\left(A \bigtimes_{k \neq j} \left(U_k^{(i-1)}\right)^*\right)^{(j)} = U_j^{(i)} \Sigma_j^{(i)} \left(V^{(i)}\right)_j^*$$

end for

$\mathcal{C} \leftarrow A \bigtimes_{k \neq j} \left(U_k^{(M)}\right)^*$

return $\mathcal{C}, \left\{U_j^{(M)}\right\}_{j \in [d]}$

Next we will show one way in the Tucker and CP rank relate. First though we note a Lemma which shows that how the mode-k product of a CP rank one tensor with a matrix U can be expressed as another rank one tensor.

Lemma 1.6.20. *Let $\mathbf{x}_j \in \mathbb{C}^{I_j}$, $U_k \in \mathbb{C}^{m_j \times I_j}$ for all $j \in [d]$ then*

$$\left(\bigcirc_{j=0}^{d-1} \mathbf{x}_j\right) \times_k U_k = \left(\bigcirc_{j=0}^{k-1} \mathbf{x}_j\right) \bigcirc U_k \mathbf{x}_k \bigcirc \left(\bigcirc_{j=k+1}^{d-1} \mathbf{x}_j\right)$$

Proof. Note that the k-mode fibers the tensor $\left(\bigcirc_{j=0}^{d-1} \mathbf{x}_j\right)$ are scalar multiples of the same vector, $\mathbf{x}_k$

That is, the k-mode fiber indexed by $(\ell_0, \ldots, \ell_{k-1}, \ell k + 1, \ldots, \ell_{d-1})$ is

$$\left(\prod_{j \neq k}^{d-1} (\mathbf{x}_j)_{\ell_j}\right) \mathbf{x}_k$$

but $\left(\prod_{j \neq k}^{d-1} (\mathbf{x}_j)_{\ell_j}\right)$ is a scalar. So the identity follows now from noting the definition of the mode-k product. (Each column of the unfolding is a scalar multiple of the same vector, scalar commutes with matrix-vector multiplication) □

Theorem 1.6.21. *If $\mathcal{A} \in \mathbb{C}^{I_0 \times \ldots I_{d-1}}$ has Tucker rank $(r_0, \ldots, r_{d-1})$ then it has CP rank of at most $\prod_{j=0}^{d-1} r_j$*

Proof. The tensor has an exact Tucker decomposition, so

$$\mathcal{A} = \mathcal{C} \bigtimes_{j=0}^{d-1} U_j^*$$

Note that we can express any tensor in the standard basis; here the standard basis for tensors is a tensor with only one non-zero entry.

So for example for some $\ell \in [r_0] \times \cdots \times [r_{d-1}]$ the associated standard basis element is

$$\bigotimes_{j=0}^{d-1} \mathbf{e}_{\ell_j}$$

where $\mathbf{e}_{\ell_j}$ is the usual standard basis vector in $\mathbb{C}^{r_j}$. Denote $\mathcal{I} = [r_0] \times \cdots \times [r_{d-1}]$. Thus

$$\mathcal{C} = \sum_{\ell \in \mathcal{I}} \mathcal{C}_\ell \left(\bigotimes_{j=0}^{d-1} \mathbf{e}_{\ell_j} \right)$$

Now use this expression in the Tucker decomposition of $\mathcal{A}$

$$\begin{aligned}
\mathcal{A} &= \mathcal{C} \bigtimes_{j=0}^{d-1} U_j^* \\
&= \left[\sum_{\ell \in \mathcal{I}} \mathcal{C}_\ell \left(\bigotimes_{j=0}^{d-1} \mathbf{e}_{\ell_j} \right) \right] \bigtimes_{j=0}^{d-1} U_j^* \\
&= \sum_{\ell \in \mathcal{I}} \mathcal{C}_\ell \left(\bigotimes_{j=0}^{d-1} U_j^* \mathbf{e}_{\ell_j} \right) \\
&= \sum_{\ell \in \mathcal{I}} \mathcal{C}_\ell \bigotimes_{j=0}^{d-1} (U_j^*)_{\ell_j}
\end{aligned}$$

where we have used Lemma 1.6.20, and denoted the ℓ_j-th column of U_j as $(U_j)_{\ell_j}$. We therefore have the sum of rank one tensors. There are $\prod_{j=0}^{d-1} r_j$ possible values for ℓ and so we have a CP decomposition of that rank. This provides an upper bound on CP rank, since the decomposition may not be optimal. □

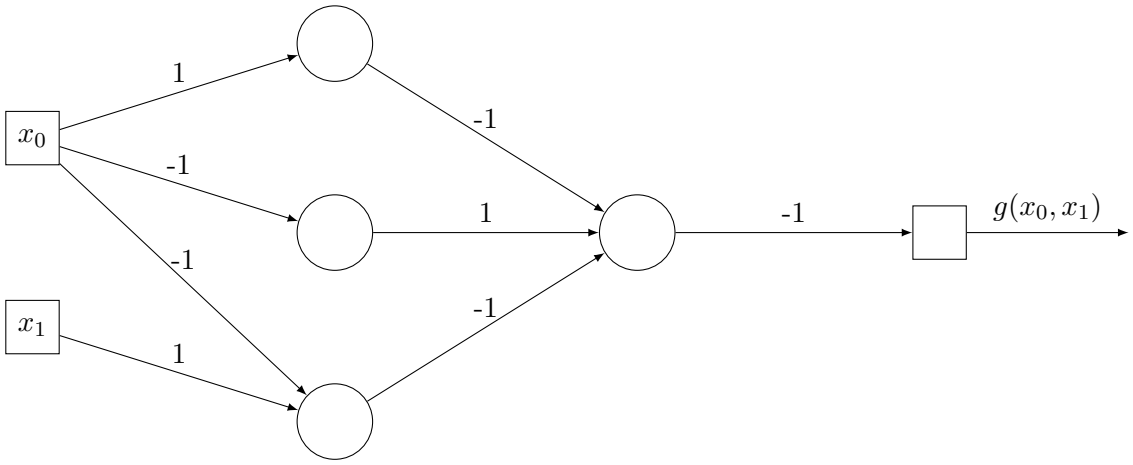


Figure 1.7: The graphical representation of the depth $L = 2$ ReLU FNN from the proof of Theorem 1.2.10 that computes $g(x_0, x_1) = \min\{0, \max\{x_0, x_1\}\}$.

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 2.4.** 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, $\mathbb{i}^2 := -1$. A **complex number** is an object of the form $z = x + iy$ 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 $z = x + iy$ is called the real part of z, and is denoted $\mathrm{Re}(z) \in \mathbb{R}$. Similarly, the number y is called the imaginary part of z, and is denoted $\mathrm{Im}(z) \in \mathbb{R}$. 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".

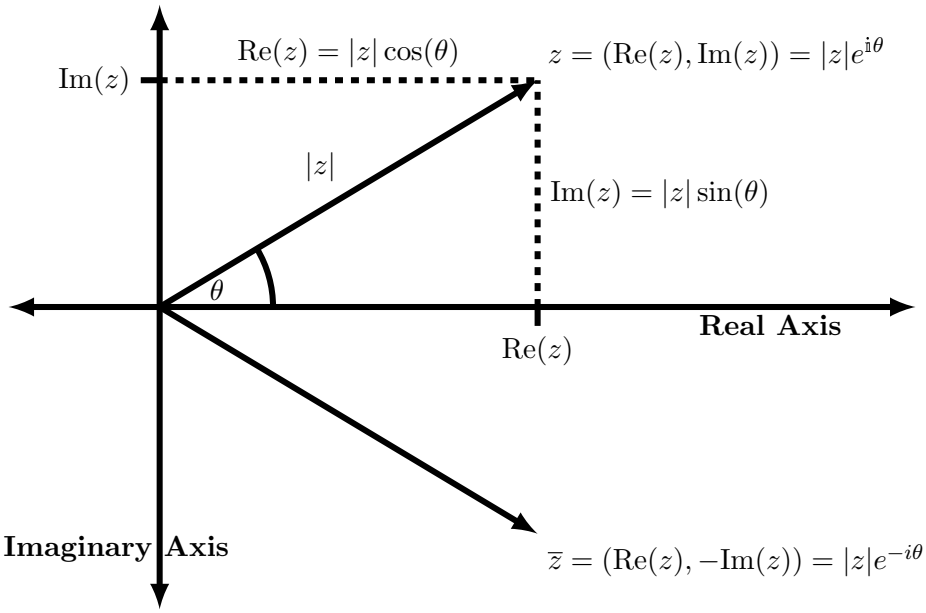


Figure 2.1: The geometry of a complex number $z \in \mathbb{C}$.

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 $(\mathrm{Re}(z), \mathrm{Im}(z)) = (x, y)$ in the complex plane. It is therefore also the length of the hypotenuse of a right triangle whose other two sides have lengths $|\mathrm{Re}(z)|$ and $|\mathrm{Im}(z)|$, and so can be computed using the Pythagorean theorem to be

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

Note that if $z \in \mathbb{R}$ so that $z = \mathrm{Re}(z)$ (i.e., if $y = 0$) then $|z| = |\mathrm{Re}(z)| = |x|$. 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 $|\mathrm{Re}(z)| \leq |z|$ and $|\mathrm{Im}(z)| \leq |z|$ 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 $(\mathrm{Re}(z), \mathrm{Im}(z)) = (x, y)$ 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

$$\begin{aligned} z_1z_2 = (x_1 + iy_1)(x_2 + iy_2) &:= x_1x_2 + \mathbb{i}x_1y_2 + \mathbb{i}y_1x_2 + \mathbb{i}^2y_1y_2 \\ &= (x_1x_2 - y_1y_2) + \mathbb{i}(x_1y_2 + y_1x_2). \end{aligned}$$

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].

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

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

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

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

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

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

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:

$$z = re^{i\theta}.$$

Stating the same formula another way, we have that

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

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

1. *Show that if $z = re^{i\theta}$, then for any $n \in \mathbb{N}$ we have $z^n = r^n\left(\cos(n\theta) + i\sin(n\theta)\right)$.*
2. *Every complex number of unit modulus can be written as $e^{i\theta}$ for some $\theta \in [0, 2\pi)$.*
3. *If $z = re^{i\theta}$ and $w = se^{i\varphi}$, then $zw = rse^{i(\theta+\varphi)}$.*

Complex Conjugation

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

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

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 $z = re^{i\theta}$. Show that $\bar{z} = re^{-i\theta}$.*

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

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

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, $z = \mathrm{e}^{\mathrm{i}\theta}$ for $\theta \in [0, 2\pi)$. By Euler's formula, this means that

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

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 $\{\mathrm{e}^{\frac{2\pi k\mathrm{i}}{n}} \mid k \in [n]\}$ 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 2.4.

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 $z_1 = r_1\mathrm{e}^{\mathrm{i}\theta_1}$ and $z_2 = r_2\mathrm{e}^{\mathrm{i}\theta_2}$ 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 $\mathrm{Im}(z_1\overline{z_2}) = r_1r_2\sin(\theta_1 - \theta_2) = 0$. 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

$$\mathrm{Re}(z_1\overline{z_2}) = r_1 r_2 \cos(\theta_1 - \theta_2) = r_1 r_2 \cos(m\pi) = -r_1 r_2 < 0.$$

This is impossible here since we have $\mathrm{Re}(z_1\overline{z_2}) = |z_1 z_2| = r_1 r_2 > 0$. 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.

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

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

Both equations are true and will be used often below.

Exercise 2.2.1. *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 write

$$AB = A\begin{pmatrix} | & & | \\ \mathbf{b}_1 & \cdots & \mathbf{b}_p \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ A\mathbf{b}_1 & \cdots & A\mathbf{b}_p \\ | & & | \end{pmatrix} \tag{2.8}$$

where $\mathbf{b}_j = B_{:,j}$ denotes the j^{th} column of B and $A\mathbf{b}_j$ is the matrix-vector multiplication described above. We can 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, the resulting matrix-matrix product $A\mathbf{v}$ agrees with the matrix-vector product definition of $A\mathbf{v}$ above.

Exercise 2.2.2. *Show that* (2.8) *holds if and only if* (2.9) *holds.*

Matrix-vector multiplication also 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 (i.e., that $A(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha A\mathbf{x} + \beta A\mathbf{y}$ for all $\alpha, \beta \in \mathbb{C}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$), and that the range of A (as a function) is the column space of A (as a matrix). That is,

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

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

Exercise 2.2.4. *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 2.2.5. *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 2.2.6. *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 2.2.7. *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 2.2.8. *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.

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 $\text{span}\{\mathbf{x}, \mathbf{y}\} \subset \mathbb{R}^n$. 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] Simply rewrite $\mathbf{x}$ and $\mathbf{y}$ in terms of an orthonormal basis of the $\text{span}\{\mathbf{x}, \mathbf{y}\}$, 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.

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.

2.2.1 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

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

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 2.2.9. *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 2.2.10. *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 2.2.11. *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 2.2.12. *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 2.2.13. *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 2.2.14. *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.

2.2.2 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 $\mathrm{span}(S)$, is the set

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

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

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

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

Exercise 2.2.16. *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).

Example 2.2.3 (The Span of S is a Linear Subspace for Every Nonempty $S \subset \mathbb{C}^n$)**.** *We need to show that*

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

As usual with set equalities of this type we will proceed by showing that both (i) $\operatorname{span}(S) \subset \operatorname{span}(\operatorname{span}(S))$*, and (ii)* $\operatorname{span}(\operatorname{span}(S)) \subset \operatorname{span}(S)$*, 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 $\operatorname{span}(\operatorname{span}(S)) \subset \operatorname{span}(S)$*, let* $\mathbf{y} \in \operatorname{span}(\operatorname{span}(S))$*. 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.*

Exercise 2.2.17. *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 set of vectors $\{\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}\} \subset \mathbb{C}^n$ 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 2.2.18. *Show that any set of vectors in $\mathbb{C}^n$ containing the zero vector is linearly dependent.*

Definition 2.2.4 (The Standard Basis Vectors of $\mathbb{C}^n$)**.** *The* **standard basis vectors of** $\mathbb{C}^n$ *are the n vectors $\{\mathbf{e}_j\}_{j\in[n]} := \{\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1},\} \subset \mathbb{C}^n$ 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 2.2.5 (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 $\alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in \mathbb{C}$ 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.

2.2.3 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 2.2.6 (The Exchange Lemma)**.** *Let $B_1, B_2 \subset \mathbb{C}^n$ be finite. Furthermore, suppose that B_2 is linearly independent, and that $\mathscr{L} := \text{span}(B_2) \subset \text{span}(B_1)$. Then $|B_2| \leq |B_1|$.*

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

Because $\mathbf{y}_0 \in \text{span}(B_1)$, there exist $\alpha_0, \ldots, \alpha_{s-1} \in \mathbb{C}$ 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, $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{s-1}\}\right)$ 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 $\mathbf{y}_1 \in \mathscr{L}$, $\mathbf{y}_1 \in \text{span}(\{\mathbf{y}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{s-1}\})$. Thus, there exists $\beta_0 \in \mathbb{C}$ and $\gamma_1, \ldots, \gamma_{s-1} \in \mathbb{C}$ 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 $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{y}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{s-1}\}\right)$ also holds.

Repeating this process $s-2$ more times we find that $\mathscr{L} \subset \text{span}\left(\{\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{s-1}\}\right)$ 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 $\mathbf{y}_0, \ldots, \mathbf{y}_{s-1}$, contradicting the fact that $\mathbf{y}_j$'s are linearly independent. Therefore, $|B_1| < |B_2|$ can't hold. □

The following corollary of Lemma 2.2.6 guarantees that any two linearly independent sets that generate the same subspace have to have the same cardinality.

Corollary 2.2.7. *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 2.2.19. *Prove Corollary 2.2.7.*

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 $\mathscr{L} = \text{span}(B)$.[3] Note that by Corollary 2.2.7 all bases of $\mathscr{L}$ must have the same cardinality. We call this cardinality the **dimension** of $\mathscr{L}$, and denote it by $\dim(\mathscr{L}) \in [n+1]$.

[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 2.2.4).

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 2.2.8 (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 2.2.20. *Let $\mathscr{L}$ be a linear subspace of $\mathbb{C}^n$. Use Lemma 2.2.6 to show that any linearly independent set of vectors $B \subset \mathscr{L}$ has cardinality $\leq n$.*

Exercise 2.2.21. *Let $\mathscr{L} \subset \mathbb{C}^n$ be a linear subspace. Prove that the dimension of $\mathscr{L}$ is at most n.*

Exercise 2.2.22. *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 $\mathscr{L}$.*

The following lemma is crucial in several later arguments.

Lemma 2.2.9. *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 2.2.18). Hence, we will be finished if we can show that $B \cup A$ is linearly independent.

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

$$\sum_{j\in[r]} \alpha_j \mathbf{b}_j + \sum_{k\in[s]} \alpha_{k+r} \mathbf{a}_k = \mathbf{0} \iff \mathscr{L} \ni \sum_{j\in[r]} \alpha_j \mathbf{b}_j = \sum_{k\in[s]} (-\alpha_{k+r}) \mathbf{a}_k \in \mathscr{K}$$
$$\iff \sum_{j\in[r]} \alpha_j \mathbf{b}_j = \mathbf{0} \text{ and } \sum_{k\in[s]} (-\alpha_{k+r}) \mathbf{a}_k = \mathbf{0}$$

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 2.2.23. *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 2.2.24. *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 $\mathcal{C}(A) \subset \mathbb{R}^m$ (which, as a reminder, is the span of the columns of A).

Exercise 2.2.25. *Show that a rank r matrix $A \in \mathbb{C}^{m\times n}$ has exactly r linearly independent columns.*

Exercise 2.2.26. *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 2.2.10. *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 2.2.27. *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 2.2.11 (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.

2.2.4 Orthonormal Bases, the Gram–Schmidt Algorithm, and the QR Decomposition of a Matrix

Algorithm 8 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 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 2.2.28. *Run Algorithm 8 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* span(S) *is indeed an orthonormal basis.*

Algorithm 8 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.

Exercise 2.2.29. *Run Algorithm 8 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 span(S) *is indeed an orthonormal basis.*

When you are finished inspecting Algorithm 8 come back here and we prove a lemma which takes a step toward showing that the set B Algorithm 8 outputs is *always* an orthonormal basis for the span of the input set S.

Lemma 2.2.12. *The set $B \subset \mathbb{C}^n$ output by Algorithm 8 is always orthonormal.*

Proof. First, we observe from Lines 3 and 8 of Algorithm 8 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 8, and noting that the vector 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 2.2.12 guarantees that the output, $B \subset \mathbb{C}^n$, of Algorithm 8 is always orthonormal, but in order for it to be a basis of $\text{span}(S)$ we also need that $\text{span}(B) = \text{span}(S)$. This is established in our next lemma.

Lemma 2.2.13. *The set $B \subset \mathbb{C}^n$ output by Algorithm 8 always satisfies span(B) = span(S).*

Proof. It suffices to show that $\text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j} = \text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j}$ 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 $\text{span}\{\mathbf{x}_0\} = \text{span}\{\mathbf{b}_0\}$ since $\mathbf{b}_0$ is a nonzero scalar multiple of $\mathbf{x}_0$ (see Line 3). Now, suppose for the sake of induction that $\text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j} = \text{span}\{\mathbf{b}_\ell\}_{l=0}^{j}$. We will prove that then $\text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j+1} = \text{span}\{\mathbf{b}_\ell\}_{l=0}^{j+1}$ must also hold in the usual two steps.

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

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

where $\alpha_{j+1} \in \mathbb{C}$ and $\mathbf{y} \in \text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j} = \text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j}$. By Lines 6 and 8 of Algorithm 8 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 $\mathbf{x}_{j+1} \in \text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}$. Therefore, $\mathbf{x} \in \text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j+1}$.

[4] Recall that only the final vector permanently selected to be $\mathbf{x}_j$ in Line 6 has its $\mathbf{y}_j \neq \mathbf{0}$. All other initially-selected/temporary $\mathbf{x}_j$ candidates have $\mathbf{y}_j = 0$, indicating that they are already in the span of $\{\mathbf{b}_\ell\}_{\ell=0}^{j-1}$.

Algorithm 9 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** $\text{span}(B) = \mathscr{L}$.
3: Pick $\mathbf{x} \in \mathscr{L} \setminus \{\mathbf{0}\}$ and initialize $B = \{\mathbf{x}/\|\mathbf{x}\|_2\}$.
4: **while** $\mathscr{L} \not\subset \text{span}(B)$ **do**
5: Pick $\mathbf{x} \in \mathscr{L} \setminus \text{span}(B)$.
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.

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

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

where $\beta_{j+1} \in \mathbb{C}$ and $\mathbf{y} \in \text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j} = \text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j}$. Again, by Lines 6 and 8 of Algorithm 8 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 $\text{span}\{\mathbf{b}_\ell\}_{\ell=0}^{j} = \text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j}$. Thus, $\mathbf{b}_{j+1} \in \text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}$ which in turn implies that $\mathbf{z} \in \text{span}\{\mathbf{x}_\ell\}_{\ell=0}^{j+1}$.

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

Combining Lemmas 2.2.12 and 2.2.13 we obtain the following theorem guaranteeing that Algorithm 8 always produces an orthonormal basis of the span of its input set, as intended.

Theorem 2.2.14. *Algorithm 8 always returns an orthonormal basis B of* $\text{span}(S) \subset \mathbb{C}^n$.

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 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 9 which is a variant of Algorithm 8 (except for subspaces). Please go and look it over.

Looking at Algorithm 9 we can see that a slightly modified version of Lemma 2.2.12 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 9 will ever terminate

Algorithm 10 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 $\mathbb{C}^n \not\subset \text{span}\left(\tilde{B}\right)$ **do**
    Pick $\mathbf{x} \in \mathbb{C}^n \setminus \text{span}\left(\tilde{B}\right)$.
    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 $\tilde{B}$.

(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 2.2.6) exactly because B will always be linearly independent (see, e.g., Exercise 2.2.20). More precisely, it will terminate after $\dim(\mathscr{L}) \leq n$ iterations (see, e.g., Exercise 2.2.22). Failing to do so would generate a contradiction. Formalizing this argument proves the following theorem.

Theorem 2.2.15. *Every nontrivial 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 10 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 10, paying special attention to its similarities and differences with Algorithm 9.

Looking at Algorithm 10 we can see that it is effectively a continuation of Algorithm 9. That is, Algorithm 10 effectively picks up where Algorithm 9 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 see that Algorithm 10 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 2.2.16. *Let $B \subset \mathbb{C}^n$ be an orthonormal basis of a 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 2.2.30. *Implement a version of Algorithm 10 in the language of your choice*[5]

[5]The language of your choice can also be "by hand".

and use it to complete the orthonormal set

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

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

Exercise 2.2.31. *Prove that every set of* n *orthonormal vectors in* $\mathbb{C}^n$ *is an orthonormal basis of* $\mathbb{C}^n$*.*

Exercise 2.2.32. *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 $A \in \mathbb{C}^{m\times n}$.

The QR Decomposition of a Matrix

Let's consider what happens when we apply Algorithm 8 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 8 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 8 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 8 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

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

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

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

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 2.2.17 (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 2.2.18. *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 2.2.17 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 2.2.7, however. For now, let us turn our attention to some implications of the QR decomposition with regard to low rank matrix compression.

Exercise 2.2.33. *Compute a QR decomposition of the matrix*

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

Verify that Q has orthonormal columns.

Exercise 2.2.34. *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 8 on the columns of the matrix. Theoretically Algorithm 8 is fantastic, but in practice a digital computer will likely turn a straightforward coding of Algorithm 8 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., [51, 17].

2.2.5 Near-Optimal Compression of Low Rank Matrices, A Recap of Gaussian Elimination, and Piles of Useful Notation

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 2.2.35. *Show that an upper triangular matrix $R \in \mathbb{C}^{r\times n}$ with $r \leq n$ will always have at least $\frac{(r-1)r}{2}$ 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., [24, 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.

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

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

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 $\text{diag}(\mathbf{v}) \in \mathbb{C}^{n\times n}$. Equivalently, $\text{diag}(\mathbf{v})$ is the $n \times n$ matrix with entries given by

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

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

Exercise 2.2.37. *Let $\mathbf{v}, \mathbf{u} \in \mathbb{C}^n$. Show that $\text{diag}(\mathbf{v})\mathbf{u} = \text{diag}(\mathbf{u})\mathbf{v} \in \mathbb{C}^n$. As a consequence, show that $\text{diag}(\mathbf{1})\mathbf{v} = \text{diag}(\mathbf{v})\mathbf{1} = \mathbf{v}$ 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

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

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

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

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

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

Exercise 2.2.38. *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 2.2.20. *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 2.2.39. *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. Atomic Permutation Matrices: 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 2.2.21. *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 2.2.40. *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 2.2.41. *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 $j_1, j_2, \ldots, j_{r-1} \in [n]$ 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

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

[6] One can revisit Example 2.2.18 see that we do generally need a permutation matrix for this to be true.

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 2.3, 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 [11]. After finishing Sections 2.2 and 2.3 the attentive reader will know everything they need to know in order to begin digesting its contents.

2.2.6 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 2.2.22 (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*

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

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*

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

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 2.2.23. *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

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

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

Exercise 2.2.42. *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 2.2.43. *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 2.2.24 (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 2.2.25. *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 $\text{span}(B_U \cup B_V) = U + V$. Let $B_U = \{\mathbf{b}_0, \ldots, \mathbf{b}_{r-1}\}$ and $B_V = \{\mathbf{d}_0, \ldots, \mathbf{d}_{s-1}\}$. Then, for any vector $\mathbf{x} \in \mathbb{C}^n$, it holds that

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

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

Corollary 2.2.26. *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

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

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 2.2.27. *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 span(U) that we started with. The idea behind projecting onto a linear subspace 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 2.2.28. *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 2.2.16. The orthonormal set $\tilde{V}$ will take the form $\tilde{V} = \{\mathbf{v}_0, \ldots, \mathbf{v}_{r-1}, \mathbf{w}_r, \ldots, \mathbf{w}_{n-1}\}$ for some $\mathbf{w}_r, \ldots, \mathbf{w}_{n-1} \in \mathbb{C}^n$. 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 $r \le \ell \le n-1$ 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 $P_U : \mathbb{C}^n \to \text{span}(U)$ only depends on $\text{span}(U)$, and not on the orthonormal set U itself.

Theorem 2.2.29. *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 2.2.28, we have that

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

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 2.2.30. *The orthogonal projection onto the x-axis of $\mathbb{C}^2$ is the function $P_{\text{x-axis}}$ 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*

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

Exercise 2.2.44. *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 2.2.45. *Let $B = \{\mathbf{b}_j\}_{j\in[r]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathscr{L} = \text{span}(B)$. 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 2.2.16. 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 2.2.31. *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, $P_{\mathscr{L}}(P_{\mathscr{L}}(\mathbf{x})) = P_{\mathscr{L}}(\mathbf{x})$ 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 2.2.46.

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 2.2.11 and Figure 2.2).

□

Theorem 2.2.31 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 2.2.45! The main contribution of Theorem 2.2.31 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).

Exercise 2.2.46. *Prove part (1) of Theorem 2.2.31.*

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

Lemma 2.2.32. *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 2.2.31 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 $P_{\mathscr{L}}(\mathbf{x}) - \mathbf{y} \neq \mathbf{0}$. □

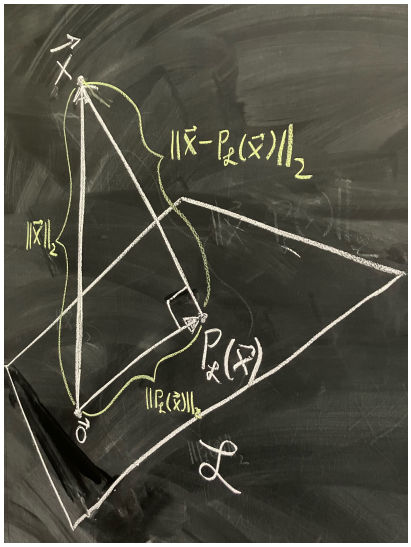


Figure 2.2: A pictorial representation of the projection of $\mathbf{x} \in \mathbb{C}^n$ onto a linear subspace $\mathscr{L} \subset \mathbb{C}^n$. Note that the right triangle whose hypotenuse is of length $\|\mathbf{x}\|_2$ will in fact be entirely contained in the two-dimensional linear subspace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

Looking at Lemma 2.2.32 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. 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.

Representing Orthogonal Projections with Matrices

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

Definition 2.2.33 (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 2.2.47. *Let $Q \in \mathbb{C}^{m \times n}$ be an orthonormal matrix. Prove that $n \leq m$.*

Exercise 2.2.48. *Let $Q \in \mathbb{C}^{m\times n}$. Show that $Q^*Q = I_n$ if and only if Q is orthonormal.*

Exercise 2.2.49. *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 $\mathscr{L} = \mathcal{C}(Q) = \text{span}(B)$ 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 2.2.50). Let $\mathbf{x} \in \mathbb{C}^n$. We have that

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

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

Lemma 2.2.34. *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 2.2.50). Using Theorem 2.2.29 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 $\mathbf{x} \in \mathbb{C}^n$. □

Exercise 2.2.50. *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 $A = B$.*

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

Theorem 2.2.35. *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.

(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

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

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

(1) $\implies$ (4): Let $U \in \mathbb{C}^{n\times n}$ be unitary. Then $\mathcal{C}(U) = \mathbb{C}^n$ (see Exercise 2.2.31). Since $\mathbb{C}^n = \text{span}\{\mathbf{e}_j\}_{j\in[n]}$ (i.e., $\mathcal{C}(I_n) = \mathbb{C}^n$) Lemma 2.2.34 tells us that $UU^* = I_nI_n^* = I_n$.

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

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

An additional consequence of Theorem 2.2.35 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 2.2.41).

Exercise 2.2.51. *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 2.2.52. *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 2.2.53. *Let $B = \{\mathbf{b}_j\}_{j\in[r]} \subset \mathbb{C}^n$ be an orthonormal basis of $\mathscr{L} = \text{span}(B)$. 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 2.2.16. 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 2.2.45.).
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 2.2.36. *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 2.2.53). 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.

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 2.2.31

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

Algorithm 11 Algorithm for (approximately) solving $A\mathbf{x} = \mathbf{b}$

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

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,

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

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. 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 11 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 11. 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 11, 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.

2.2.7 The Four Fundamental Linear Subspaces of a Matrix, and The Spectral Theorem for Hermitian Matrices

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 2.2.54. *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 2.2.37. *Let $A \in \mathbb{C}^{m\times n}$. It's always the case that $r = \text{rank}(A) = \text{rank}(A^*)$.*

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

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

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 2.2.6) 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 and A^* interchanged 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 2.2.37 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.

Algorithm 12 ALGORITHM FOR COMPUTING AN ORTHONORMAL BASIS OF $\mathcal{N}(A)$

1: **Input: A rank r matrix $A \in \mathbb{C}^{m\times n}$.**
2: **Output: An orthonormal basis of A's null space $\mathcal{N}(A) \subset \mathbb{C}^n$.**
3: Compute a QR decomposition of A^*, so that $A^* = QR$. Note that $\mathcal{C}(A^*) = \mathcal{C}(Q)$.
4: 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 $\mathcal{C}(Q)^\perp = \mathcal{C}(A^*)^\perp = \mathcal{N}(A)$.
5: Return S.

Lemma 2.2.38. *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 2.2.36 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

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

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 2.2.38 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 8 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 10. The set S will be an orthonormal basis of $\mathcal{N}(A)$ of size $n - \text{rank}(A)$. See Algorithm 12 for pseudocode.

Exercise 2.2.55. *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 $\dim(\mathcal{N}(A^*)) = m - r$.*

The next lemma will be important soon in Section 2.3. We will prove it here since it depends crucially on our recent revelations regarding null spaces.

Lemma 2.2.39. *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 2.2.56. *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

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

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 2.3.

Definition 2.2.40. *A matrix $A \in \mathbb{C}^{n\times n}$ is called **Hermitian** if $A = A^*$.*

Exercise 2.2.57. *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 2.2.58. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian. Show that all entries on A's diagonal are real numbers.*

Exercise 2.2.59. *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 2.2.41. *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 2.2.42. *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 $A - \lambda I_n \in \mathbb{C}^{n\times n}$,

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

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 2.2.43. *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 2.2.60. *Prove that every matrix $A \in \mathbb{C}^{n\times n}$ with rank $< n$ has at least one nontrivial eigenspace. What is it?*

Exercise 2.2.61. *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 2.2.44. *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 $\lambda_0, \ldots, \lambda_{m-1} \in \mathbb{R}$. Lemma 2.2.44 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 $\mathcal{N}(A - \lambda_j I_n)$ of A, then

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

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., [27, Chapter 2] or [23, Chapter 14] for corroborating evidence).

Fact 2.2.45. *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

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

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

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

This computation together with Lemma 2.2.42, Lemma 2.2.44, and Fact 2.2.45 prove the following theorem (see also, e.g., Theorem 2.5.6 in [27]).

Theorem 2.2.46 (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*

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

Exercise 2.2.62. *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 2.2.46 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 2.3.

Recall from our definition of atomic permutation matrices $P(j,k) \in \mathbb{C}^{n\times n}$ (see Example 2.2.21 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 2.2.41 and Theorem 2.2.35). 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,

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

Example 2.2.47. *Let* $P(0,2) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \in \mathbb{C}^{3\times 3}$. *We can see that*

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

Using these facts about atomic permutation matrices together with Theorem 2.2.46, 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

$$\begin{aligned} A = \ U\ \text{diag}(\lambda_0,\ldots,\lambda_{n-1})\ U^* \ &= \ U(PP^*)\ \text{diag}(\lambda_0,\ldots,\lambda_{n-1})\ (PP^*)U^* \\ &= (UP)(P\ \text{diag}(\lambda_0,\ldots,\lambda_{n-1})\ P)(P^*U^*) \\ &= (UP)\ \text{diag}(\tilde{\lambda}_0,\ldots,\tilde{\lambda}_{n-1})\ (UP)^*, \end{aligned}$$

where $\tilde{\lambda}_0,\ldots,\tilde{\lambda}_{n-1}$ is a permutation of $\lambda_0,\ldots,\lambda_{n-1} \in \mathbb{R}$ 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 2.2.51).

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 $\text{diag}(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_{n-1})$ in block form as well by

$$\text{diag}(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_{n-1}) = \begin{pmatrix} D & \mathbf{0} \\ \mathbf{0}^* & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

where $D = \text{diag}(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_{n-2}) \in \mathbb{R}^{(n-1) \times (n-1)}$ 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 2.2.48 (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

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

We end our review of basic linear algebra here by noting that Theorem 2.2.46 and Corollary 2.2.48 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 2.2.46 effectively replaces it with). Theorem 2.2.46 and Corollary 2.2.48 do have one major flaw, however. They only apply to one very special type of square matrix! In the next section we will remove this flaw by developing a generalization of these Hermitian matrix decompositions that applies to all (even rectangular) matrices.

2.3 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 [49]. 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 2.3.1. *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*

$$\mathbf{h}_j := \begin{cases} \mathbf{0} & \text{if } s_j = 0 \\ \frac{1}{s_j} A\mathbf{w}_j \in \mathbb{C}^m & \text{if } s_j \neq 0 \end{cases}. \tag{2.16}$$

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

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

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

Proof. We have that

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

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

Lemma 2.3.1 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 (2.16) 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 2.3.1.

As we shall see, choosing $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ in Lemma 2.3.1 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 2.2.45 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 2.2.7. 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 2.2.38 and 2.2.39). Second, A is rank r if and only if A^*A is rank r (see Theorem 2.2.37 and Lemma 2.2.39). 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 2.3.1. *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 $A^*A \in \mathbb{C}^{n\times n}$. Prove that exactly r of the orthonormal eigenvectors of A^*A in $\{\mathbf{w}_j\}_{j\in[n]}$ will be associated with nonzero eigenvalues. Suppose, w.l.g., that these r orthonormal eigenvectors of A^*A are $\{\mathbf{w}_j\}_{j\in[r]}$. Show that they are an orthonormal basis of $\mathcal{C}(A^*)$.*

Exercise 2.3.2. *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 $\{\mathbf{w}_j\}_{j\in[n]}$. Suppose, w.l.g., that these $n-r$*

orthonormal eigenvectors of A^*A 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 2.3.1 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 (2.16).

Lemma 2.3.2. *Let $A \in \mathbb{C}^{m\times n}$ be rank r. Choose $\{\mathbf{w}_j\}_{j\in[n]} \subset \mathbb{C}^n$ in Lemma 2.3.1 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* (2.16) *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 $j = r, \ldots, n-1$.*

Proof. Exactly r of the $\mathbf{h}_j$ vectors defined in (2.16) will be nonzero by Exercise 2.3.2. 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 (2.16). Thus, to finish the proof it suffices by Exercise 2.2.32 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 $0 \leq \ell < r$. Considering the inner product of any two nonzero $\mathbf{h}_j$ vectors from (2.16) 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 2.3.3. *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 2.3.1 results in exactly r orthonormal $\mathbf{h}_j$ vectors in* (2.16)*. Prove that every $\mathbf{w}_j$ must then be an eigenvector of $A^*A \in \mathbb{C}^{n\times n}$.*

Lemma 2.3.2 combined with Exercise 2.3.3 imply that there is essentially only one way to apply Lemma 2.3.1 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

$$\begin{aligned} AV &= \begin{pmatrix} | & | & & | \\ A\mathbf{w}_0 & A\mathbf{w}_1 & \cdots & A\mathbf{w}_{n-1} \\ | & | & & | \end{pmatrix} \\ &= \begin{pmatrix} | & | & & | & | & | & | \\ s_0\mathbf{h}_0 & s_1\mathbf{h}_1 & \cdots & s_{r-1}\mathbf{h}_{r-1} & \mathbf{0} & \cdots & \mathbf{0} \\ | & | & & | & | & | & | \end{pmatrix} \in \mathbb{C}^{m\times n} \qquad (2.17) \\ &= \underbrace{\begin{pmatrix} | & | & & | & | & & | \\ \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_{r-1} & \mathbf{u}_r & \cdots & \mathbf{u}_{m-1} \\ | & | & & | & | & & | \end{pmatrix}}_{\in\mathbb{C}^{m\times m}} \underbrace{\begin{pmatrix} \operatorname{diag}(s_0,\ldots,s_{r-1}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0}\ \mathbf{0} & \cdots & \mathbf{0}\ \mathbf{0}\ \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}}_{\in\mathbb{C}^{m\times n}} \\ &= U\Sigma, \end{aligned}$$

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

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

Example 2.3.3. *To help the reader digest the abstract computation in* (2.17) *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 2.3.1, 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 (2.17) 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 2.3.4. *Repeat the calculation in Example 2.3.3 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 2.3.4 (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.*[7]

Exercise 2.3.5. *Let $A \in \mathbb{C}^{m\times n}$ have the full SVD $A = U\Sigma V^*$. Set $r = \operatorname{rank}(A)$. Show that*

$$A = \sum_{j\in[r]} \sigma_j(A)\mathbf{u}_j\mathbf{v}_j^* \tag{2.18}$$

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 2.2.50.)

One can now prove the following corollary from Theorem 2.3.4 via an argument analogous to the one used to derive Corollary 2.2.48 from Theorem 2.2.46 (or, alternatively, by using (2.18) from Exercise 2.3.5 to build the new factorization more directly).

[7] 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.

Corollary 2.3.5 (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. *$\Sigma = \text{diag}\left(\sigma_0(A), \ldots, \sigma_{r-1}(A)\right) \in [0,\infty)^{r\times r}$ 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 2.3.6. *Prove Corollary 2.3.5.*

However one proves Theorem 2.3.4 and Corollary 2.3.5, 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^*.

2.3.1 The Relationship Between any Valid SVD of A and 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., [23, Chapter 10]), and always nonnegative real numbers (see Exercise 2.2.62), 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 2.3.7. *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^{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 2.3.6. *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^{th} singular value of A (i.e., $\sigma_j = |\lambda|$),*
   - *the j^{th} column of V, $\mathbf{v}_j \in \mathbb{C}^n$, is an eigenvector of A associated with λ, and*
   - *the j^{th} column of $U = \operatorname{sign}(\lambda)\mathbf{v}_j$.*

Exercise 2.3.8. *Prove Theorem 2.3.6.*

Exercise 2.3.9. *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 2.3.10. *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 2.3.6 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 [51, Lecture 31] or [17, 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, 31]).

2.3.2 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 an inverse, denoted by $A^\dagger : C(A) \to \mathcal{C}(A^*)$, that's called the **Moore–Penrose (or, pseudo)inverse of** A. Furthermore, a factorization of $A^\dagger \in \mathbb{C}^{n\times m}$ can be computed easily using 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 $\Sigma = \text{diag}\left(\sigma_0(A), \ldots, \sigma_{r-1}(A)\right) \in [0,\infty)^{r\times r}$ is invertible (due to $\sigma_0(A) \geq \cdots \geq \sigma_{r-1}(A) > 0$). One can now see that

$$A^\dagger = V\Sigma^{-1}U^* \tag{2.19}$$

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 2.2.31). 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 2.3.6. Hence, $A^\dagger : C(A) \to \mathcal{C}(A^*)$ from (2.19) 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 2.3.11. *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 (2.19) satisfies $AA^\dagger = P_{\mathcal{C}(A)}$.*

Exercise 2.3.12. *Suppose that $A \in \mathbb{C}^{n\times n}$ is full rank (so that $\text{rank}(A) = n$). 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 2.3.13. *Suppose that $A \in \mathbb{C}^{n\times n}$ is full rank (so that $\text{rank}(A) = n$). 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 $j \in [n]$.*

2.3.3 Singular Values, Matrix Norms, and Some Singular Value Inequalities

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 2.2.1 are closely related to the singular values of a given matrix.

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

$$\begin{aligned}\|A\|_{\mathrm{F}}^2 &= \|U\Sigma V^*\|_{\mathrm{F}}^2 = \sum_{j\in[n]} \|\,(U\Sigma V^*)_{:,j}\,\|_2^2 = \sum_{j\in[n]} \|U\,(\Sigma V^*)_{:,j}\,\|_2^2 \\ &= \sum_{j\in[n]} \|\,(\Sigma V^*)_{:,j}\,\|_2^2 = \|\Sigma V^*\|_{\mathrm{F}}^2\end{aligned}$$

by Exercise 2.2.52 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

$$\|A\|_{\mathrm{F}}^2 = \|V\Sigma^*\|_{\mathrm{F}}^2 = \sum_{j\in[m]} \|V\,(\Sigma^*)_{:,j}\,\|_2^2 = \sum_{j\in[m]} \|\Sigma^*_{:,j}\|_2^2 = \sum_{j\in[q]} (\sigma_j(A))^2. \tag{2.20}$$

We will see that (2.20) has several important implications in later sections.

Exercise 2.3.14. *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 $A \in \mathbb{C}^{m\times n}$.*

The (ℓ^2,ℓ^2)-Operator Norm

Given $A \in \mathbb{C}^{m\times n}$ recall that $\|A\|_{2\to 2} = \max\limits_{\mathbf{x}\in\mathbb{C}^n \text{ s.t. } \|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2$. 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

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

by Exercise 2.2.52 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 2.2.11). 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,

$$\|A\|_{2\to 2} = \sigma_0(A). \tag{2.21}$$

We will see that (2.21) also has several important implications in later sections.

Exercise 2.3.15. *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 2.3.16. *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?*

Some Singular Value Inequalities

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 (2.20)) 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 2.3.7 (See Theorem 3.3.16 in [26])**.** *Let $A, B \in \mathbb{C}^{m\times n}$ and $q = \min\{m, n\}$. Then*

(a) $\sigma_{j+k}(A + B) \leq \sigma_j(A) + \sigma_k(B)$, *and*

(b) $\sigma_{j+k}(AB^*) \leq \sigma_j(A)\,\sigma_k(B)$

for all $j, k \in [q]$ such that $j + k \in [q]$. In particular,

(c) $|\sigma_j(A + B) - \sigma_j(A)| \leq \sigma_0(B) \quad \forall j \in [q]$, *and*

(d) $\sigma_j(AB^*) \leq \sigma_j(A)\,\sigma_0(B) \qquad \forall j \in [q]$.

Exercise 2.3.17. *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 2.3.18. *Use parts* (a) *and* (b) *of Theorem 2.3.7 to prove parts* (c) *and* (d)*.*

Looking at Theorem 2.3.7 (c) one can see that if $B = E$ has a small largest singular value, $\sigma_0(E)$, then we will indeed have $\sigma_j(A+E) \approx \sigma_j(A)$ for all $j \in [q]$. Furthermore, one can also use these inequalities 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 extremely.

Example 2.3.8. *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 2.3.16) 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 will be useful in later chapters.

Lemma 2.3.9 (A Slight Generalization of Theorem 2.3.7 Part (d))**.** *Let* $A \in \mathbb{C}^{m\times n}$ *and* $B \in \mathbb{C}^{n\times p}$*. Then,*

$$\sigma_j(AB) \leq \min\{\sigma_j(A)\sigma_0(B), \sigma_j(B)\sigma_0(A)\} \quad \forall j \in [\min\{n,m,p\}].$$

Proof. Suppose, without loss of generality, that $m \leq p$ (else, we may instead apply the argument below to $\sigma_j((AB)^*) = \sigma_j(B^*A^*)$ using that $\sigma_j(AB) = \sigma_j((AB)^*)$). 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

$$\sigma_j(QA) = \sigma_j(A) \text{ and } \sigma_j(QAB) = \sigma_j(AB) \quad \forall j \in [\min\{m,n\}] = [\min\{n,m,p\}].$$

Applying part (d) of Theorem 2.3.7 we can now see that both

$$\sigma_j(AB) = \sigma_j(QAB) \leq \sigma_j(QA)\sigma_0(B^*) = \sigma_j(A)\sigma_0(B)$$

and

$$\sigma_j(AB) \;=\; \sigma_j(QAB) \;=\; \sigma_j(B^*(QA)^*) \;\leq\; \sigma_j(B^*)\,\sigma_0(QA) \;=\; \sigma_j(B)\,\sigma_0(A)$$

hold. The result follows. □

Lemma 2.3.10. *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,

$$\sigma_j(AB) \geq \sigma_r(AU)\,\sigma_j(B) \qquad \forall j \in [r].$$

Proof. Let $j \in [r]$. Noting that P is unitary since $r = n$, we can see that

$$\sigma_j(B) \;=\; \sigma_j(\Sigma V^*) \;=\; \sigma_j\left(P\tilde{\Sigma}^{-1}\tilde{\Sigma}P^*\Sigma V^*\right) \;\leq\; \sigma_0\left(P\tilde{\Sigma}^{-1}\right)\cdot\sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right)$$

by Lemma 2.3.9. Furthermore, $\sigma_0\left(P\tilde{\Sigma}^{-1}\right) = \sigma_0\left(\tilde{\Sigma}^{-1}\right) = \frac{1}{\sigma_r(\tilde{\Sigma})} = \frac{1}{\sigma_r(AU)}$. Hence,

$$\sigma_j(B) \;\leq\; \frac{\sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right)}{\sigma_r(AU)} \implies \sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right) \geq \sigma_r(AU)\,\sigma_j(B). \tag{2.22}$$

Finally, since $m \geq r = n$ we can see that $Q^*Q = I_n$ so that

$$\begin{aligned}\sigma_j\left(\tilde{\Sigma}P^*\Sigma V^*\right) &= \sigma_j\left(Q^*Q\tilde{\Sigma}P^*\Sigma V^*\right) \;=\; \sigma_j(Q^*AU\Sigma V^*) \;=\; \sigma_j(Q^*AB)\\ &\leq \sigma_j(AB)\,\sigma_0(Q^*) \;=\; \sigma_j(AB)\end{aligned} \tag{2.23}$$

by Lemma 2.3.9. Combing (2.22) and (2.23) now finishes the proof. □

Though perturbation bounds for singular values such as those in Theorem 2.3.7 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., [47, 48] to get a good overview of these results.

2.3.4 Optimal Low-Rank Approximation

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 2.2.5 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 2.3.8 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 2.2.5, 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 2.3.5). We claim that

$$A_s := \sum_{j\in[s]} \sigma_j(A)\mathbf{u}_j\mathbf{v}_j^* \tag{2.24}$$

is then an optimal rank s approximation to A with respect to both the Frobenius and the (ℓ^2, ℓ^2)-operator norms.

Exercise 2.3.19. *Let $A \in \mathbb{C}^{m\times n}$, $q = \min\{m, n\}$, and $A_s \in \mathbb{C}^{m\times n}$ be as in (2.24). Show that $\sigma_j(A - A_s) = \sigma_{j+s}(A)$ for all $j \in [q-s]$, and that $\sigma_j(A - A_s) = 0$ for all $q - s \le j < q$.*

Optimality of A_s in the Frobenius Norm

Observe that for the Frobenius norm we have

$$\|A - A_s\|_{\mathrm{F}}^2 \;=\; \left\| \sum_{j=s}^{q} \sigma_j(A) \mathbf{u}_j \mathbf{v}_j^* \right\|_{\mathrm{F}}^2 \;=\; \sum_{j=s}^{q-1} \sigma_j^2(A) \tag{2.25}$$

by (2.20) and Exercise 2.3.19. The next theorem shows that this approximation error is minimal.

Theorem 2.3.11. *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 (2.24). 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 2.3.4 since B is rank s. Thus, Theorem 2.3.7 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, (2.20) and (2.25) 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. □

Optimality of A_s in the (ℓ^2, ℓ^2)-Operator Norm

Observe that for the (ℓ^2, ℓ^2)-operator norm we have

$$\|A - A_s\|_{2\to 2} \;=\; \left\| \sum_{j=s}^{q} \sigma_j(A) \mathbf{u}_j \mathbf{v}_j^* \right\|_{2\to 2} \;=\; \sigma_s(A) \tag{2.26}$$

by (2.21) and Exercise 2.3.19. The next theorem shows that this approximation error is also minimal.

Theorem 2.3.12. *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 (2.24). 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, $\mathscr{L} = \text{span}\{V_{:,0}, \ldots, V_{:,s}\} \subset \mathbb{C}^n$ has dimension $s + 1$. Also, we know that $\mathcal{C}(R^*)^\perp = \mathcal{N}(R)$ has dimension $n - s$ by (the discussion around) Lemma 2.2.38. Hence, it must be the case that

$$\text{span}\{V_{:,0}, \ldots, V_{:,s}\} \cap \mathcal{N}(R)$$

is a linear subspace of $\mathbb{C}^n$ of dimension at least 1 by Exercise 2.2.24. Thus, there exists $\mathbf{n} \in \text{span}\{V_{:,0}, \ldots, V_{:,s}\} \cap \mathcal{N}(R)$ 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 $\alpha_0, \ldots, \alpha_s \in \mathbb{C}$ with $\|\boldsymbol{\alpha}\|_2 = 1$ (recall Theorem 2.2.11), 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 (2.26). □

2.4 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.

Definition 2.4.1 (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* $A = \text{Toep}_{m,n}(\mathbf{a})$*. 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* $A = \text{Toep}_{m,n}(\mathbf{a})$*. We will also define* $\text{Toep}_n(\mathbf{a}) := \text{Toep}_{n,n}(\mathbf{a})$ *in the case of square matrices.*

Example 2.4.2. *The Toeplitz matrix* $A \in \mathbb{C}^{3\times 4}$ *generated by* $\mathbf{a} \in \mathbb{C}^6$ *is*

$$\text{Toep}_{3,4}(\mathbf{a}) = \begin{pmatrix} a_2 & a_3 & a_4 & a_5 \\ a_1 & a_2 & a_3 & a_4 \\ a_0 & a_1 & a_2 & a_3 \end{pmatrix}.$$

The Toeplitz matrix $A \in \mathbb{C}^{4\times 3}$ *generated by* $\mathbf{a} \in \mathbb{C}^6$ *is*

$$\text{Toep}_{4,3}(\mathbf{a}) = \begin{pmatrix} a_3 & a_4 & a_5 \\ a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 \\ a_0 & a_1 & a_2 \end{pmatrix}.$$

Note that the entries of $\mathbf{a}$ *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 2.4.1. *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 2.4.2. *Show that* A *is Toeplitz if and only if* A^* *is Toeplitz.*

Exercise 2.4.3. *Given* $\mathbf{a} \in \mathbb{C}^n$ *let* $\text{Reverse}(\mathbf{a}) \in \mathbb{C}^n$ *be the vector with entries given by*

$$(\text{Reverse}(\mathbf{a}))_j = \overline{a_{n-1-j}}.$$

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* $\text{Reverse}(\mathbf{a})$*.*

Definition 2.4.3 (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),* $\sigma(W\mathbf{x} + \mathbf{b})$*, 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 2.4.2 and 2.4.3. 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".

2.4.1 Circulant and Toeplitz Matrices

As we will see later in Section 2.4.2, the following special class of square Toeplitz matrices is crucial to realizing fast matrix-vector multiplication algorithms for more arbitrary Toeplitz matrices.

Definition 2.4.4 (Circulant Matrix)**.** *The* ***circulant matrix generated by a vector*** $\mathbf{v} \in \mathbb{C}^n$ *is the matrix* $\text{circ}(\mathbf{v}) \in \mathbb{C}^{n\times n}$ *defined by*

$$(\text{circ}(\mathbf{v}))_{j,k} = \mathbf{v}_{(j-k) \bmod n}.$$

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* $A = \text{circ}(\mathbf{v})$*. Herein "*$j \bmod n$*" 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, it is the unique value* $r \in [n] = \{0, 1, \ldots, n-1\}$ *such that* $\exists k \in \mathbb{Z}$ *satisfying* $j = r + kn$*).*

Example 2.4.5. *The circulant matrix* $A \in \mathbb{C}^{4\times 4}$ *generated by* $\mathbf{v} \in \mathbb{C}^4$ *is*

$$\text{circ}(\mathbf{v}) = \begin{pmatrix} v_0 & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_3 & v_2 \\ v_2 & v_1 & v_0 & v_3 \\ v_3 & v_2 & v_1 & v_0 \end{pmatrix}.$$

Note that this matrix is also Toeplitz due to the fact that it's constant along its diagonals (recall Exercise 2.4.1).

Exercise 2.4.4. *Show that every circulant matrix is also Toeplitz.*

Exercise 2.4.5. *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 $\mathbf{a}$, $\text{Toep}_n(\mathbf{a}) \in \mathbb{C}^{n\times n}$, with entries given by

$$(\text{Toep}_n(\mathbf{a}))_{j,k} = a_{(n-1)-(j-k)}. \tag{2.27}$$

Now let $\mathbf{c} \in \mathbb{C}^{2n}$ be defined by $\mathbf{c}^T = (a_{n-1}, a_{n-2}, \ldots, a_0, 0, a_{2n-2}, \ldots, a_n)$ so that

$$c_\ell = \begin{cases} a_{n-1-\ell} & 0 \le \ell \le n-1 \\ 0 & \ell = n \\ a_{3n-1-\ell} & n+1 \le \ell \le 2n-1 \end{cases} \tag{2.28}$$

for all $\ell \in [2n]$. Then, the circulant matrix $\text{circ}(\mathbf{c}) \in \mathbb{C}^{2n\times 2n}$ will always take the block form

$$\text{circ}(\mathbf{c}) = \begin{pmatrix} \text{Toep}_n(\mathbf{a}) & A \\ A & \text{Toep}_n(\mathbf{a}) \end{pmatrix} \in \mathbb{C}^{2n\times 2n}, \tag{2.29}$$

where $A \in \mathbb{C}^{n\times n}$.

Example 2.4.6. *Let* $\mathbf{a} \in \mathbb{C}^3$. *The* 2×2 *Toeplitz matrix generated by* $\mathbf{a}$ *is*

$$\text{Toep}_2(\mathbf{a}) = \begin{pmatrix} a_1 & a_2 \\ a_0 & a_1 \end{pmatrix}.$$

If we form the vector $\mathbf{c} \in \mathbb{C}^4$ *defined by* $\mathbf{c}^T = (a_1, a_0, 0, a_2)$ *then*

$$\text{circ}(\mathbf{c}) \;=\; \begin{pmatrix} a_1 & a_2 & 0 & a_0 \\ a_0 & a_1 & a_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ a_2 & 0 & a_0 & a_1 \end{pmatrix} \;=\; \begin{pmatrix} \text{Toep}_2(\mathbf{a}) & A \\ A & \text{Toep}_2(\mathbf{a}) \end{pmatrix} \in \mathbb{C}^{4\times 4}.$$

The following lemma guarantees the upper-left $n \times n$ block of $\text{circ}(\mathbf{c}) \in \mathbb{C}^{2n\times 2n}$ is indeed always $\text{Toep}_n(\mathbf{a}) \in \mathbb{C}^{n\times n}$ as claimed above in (2.29).

Lemma 2.4.7. *Let* $\mathbf{a} \in \mathbb{C}^{2n-1}$. *Build* $\mathbf{c} \in \mathbb{C}^{2n}$ *from* $\mathbf{a}$ *entry-wise via* (2.28). *Then,* $(\text{circ}(\mathbf{c}))_{j,k} = (\text{Toep}_n(\mathbf{a}))_{j,k}$ *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

$$(\text{circ}(\mathbf{c}))_{j,k} \;=\; c_{(j-k) \bmod n} \;=\; c_{j-k} \;=\; a_{n-1-(j-k)},$$

and if $-(n-1) \leq j-k < 0$ we have that

$$(\text{circ}(\mathbf{c}))_{j,k} \;=\; c_{(j-k) \bmod n} \;=\; c_{2n+(j-k)} \;=\; a_{3n-1-(2n+(j-k))} \;=\; a_{n-1-(j-k)}.$$

Thus, $(\text{circ}(\mathbf{c}))_{j,k} \;=\; (\text{Toep}_N(\mathbf{a}))_{j,k} \;=\; a_{(n-1)-(j-k)}$ for all $j, k \in [n]$ by (2.27). □

Exercise 2.4.6. *Use Lemma 2.4.7 together with Exercise 2.4.5 to show that* (2.29) *holds for all* $n \in \mathbb{N}$.

Computationally, observe that via (2.29) we have for any $\mathbf{v} \in \mathbb{C}^n$ that

$$\text{circ}(\mathbf{c}) \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \text{Toep}_n(\mathbf{a}) & A \\ A & \text{Toep}_n(\mathbf{a}) \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \text{Toep}_n(\mathbf{a})\mathbf{v} \\ A\mathbf{v} \end{pmatrix}. \tag{2.30}$$

Thus, we can always recover $\text{Toep}_n(\mathbf{a})\mathbf{v}$ from $\text{circ}(\mathbf{c}) \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$ 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 2.4.3.

2.4.2 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 2.4.7. *Show that* $(f_n)^{kn} = f_n^{kn} = 1$ *for all* $k \in \mathbb{Z}$.

Exercise 2.4.8. *Show that* $(f_n)^k = f_n^k \neq 1$ *for all nonzero* $k \in [n]$.

Exercise 2.4.9. *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* $j, \omega \in \mathbb{Z}$.[8]

Exercise 2.4.10. *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 2.4.11). This basis is called the **discrete Fourier basis of** $\mathbb{C}^n$.

[8] Let $j \in \mathbb{Z}$. Recall that "$j \bmod n$" denotes the unique integer $r \in [n]$ satisfying $j = r + k \cdot n$ for some $k \in \mathbb{Z}$.

Example 2.4.8. *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 2.4.7 and 2.4.8. 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 2.4.11. *Prove that the DFT matrix,* F*, is unitary. (HINT: Recall Theorem 2.2.35.)*

Exercise 2.4.12. *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.*

The **Discrete Fourier Transform (DFT)** of a vector $\mathbf{v} \in \mathbb{C}^n$ is simply

$$\widehat{\mathbf{v}} := F\mathbf{v} \tag{2.31}$$

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 $\omega \in [n] = \{0, \ldots, n-1\} \subset \mathbb{N}$. Similarly, the **Inverse Discrete Fourier Transform (IDFT)** of a vector $\mathbf{v} \in \mathbb{C}^n$ is

$$\widehat{\mathbf{v}}^{-1} := F^{-1}\mathbf{v} = F^*\mathbf{v}.$$

As we shall see, the DFT walks hand in hand with our next definition.

Exercise 2.4.13. *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*

$$v_j = \begin{cases} u_{pj/n} & \text{if } j \equiv 0 \bmod (n/p) \\ 0 & \text{else} \end{cases},$$

and let $\mathbf{w} \in \mathbb{C}^n$ *be another longer vector with entries given by* $w_j = u_{j \bmod p}$*. 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 2.4.14. *Let $a, b, c \in [n]$ be such that a is invertible modulo n.*[9] *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, $\mathbf{u} \star \mathbf{v} = \mathrm{circ}(\mathbf{v})\mathbf{u}$ for all $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$.

Example 2.4.9. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{C}^4$. Then,*

$$\mathbf{u} \star \mathbf{v} \;=\; \begin{pmatrix} \sum_{j=0}^{3} u_j \; v_{-j \bmod n} \\ \sum_{j=0}^{3} u_j \; v_{1-j \bmod n} \\ \sum_{j=0}^{3} u_j \; v_{2-j \bmod n} \\ \sum_{j=0}^{3} u_j \; v_{3-j \bmod n} \end{pmatrix} \;=\; \begin{pmatrix} v_0 & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_3 & v_2 \\ v_2 & v_1 & v_0 & v_3 \\ v_3 & v_2 & v_1 & v_0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} \;=\; \mathrm{circ}(\mathbf{v})\mathbf{u}.$$

The discrete convolution has the following useful relationship with the discrete Fourier transform.

Theorem 2.4.10. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$. Then*

$$\widehat{(\boldsymbol{u} \star \boldsymbol{v})}_\omega = \sqrt{n}\, \widehat{u}_\omega \widehat{v}_\omega \tag{2.32}$$

holds for all $\omega \in [n]$.

Proof: To obtain (2.32) 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

$$\widehat{(\mathbf{u} \star \mathbf{v})}_\omega \;=\; \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} u_j \, f_n^{\omega \cdot j} \left(\sum_{k=0}^{n-1} v_{(k-j) \bmod n} \, f_n^{\omega \cdot (k-j)} \right) \;=\; \sqrt{n}\, \widehat{u}_\omega \widehat{v}_\omega.$$

[9] 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., [41, Theorem 2.8]).

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 2.4.9). $\square$

Exercise 2.4.15. *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 2.4.10 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 2.4.10 now directly implies that

$$\mathbf{u} \star \mathbf{v} = \sqrt{n}\, \widehat{\widehat{\mathbf{u}} \odot \widehat{\mathbf{v}}}^{-1} = \sqrt{n}\, F^* (F\mathbf{u} \odot F\mathbf{v}). \tag{2.33}$$

Note that the last expression of (2.33) 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 2.4.3.

The following additional fact relating IDFT matrices to circulant matrices is closely related to Theorem 2.4.10: 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 2.4.11. *Let* $\mathbf{v} \in \mathbb{C}^n$. *Every column of* $F^* \in \mathbb{C}^{n \times n}$ *is an eigenvector of* $\mathrm{circ}(\mathbf{v})$.

Proof. Let $\mathbf{u} = F^* \mathbf{e}_j \in \mathbb{C}^n$ be the j^{th} column of F^*. By (2.33),

$$\begin{aligned} \mathrm{circ}(\mathbf{v})\mathbf{u} = \mathbf{u} \star \mathbf{v} &= \sqrt{n}\, F^*(F\mathbf{u} \odot F\mathbf{v}) = \sqrt{n}\, F^*((FF^*\mathbf{e}_j) \odot F\mathbf{v}) \\ &= \sqrt{n}\, F^*(\mathbf{e}_j \odot \widehat{\mathbf{v}}) = \sqrt{n}\, F^*(\widehat{v}_j \mathbf{e}_j) = \sqrt{n}\, \widehat{v}_j \mathbf{u} \end{aligned}$$

Thus, the j^{th} column of F^* is an eigenvector of $\mathrm{circ}(\mathbf{v})$ with eigenvalue $\sqrt{n}\, \widehat{v}_j$. $\square$

Exercise 2.4.16. *Let* $\mathbf{v} \in \mathbb{C}^n$. *Show that* $\mathrm{circ}(\mathbf{v}) \in \mathbb{C}^{n \times n}$ *is invertible if and only if* $\widehat{v}_\omega \neq 0$ *for all* $\omega \in [n]$.

Exercise 2.4.17. *Order the Fourier coefficients of* $\mathbf{v} \in \mathbb{C}^n$ *by magnitude so that*

$$|\widehat{v}_{\omega_0}| \geq |\widehat{v}_{\omega_2}| \geq \cdots \geq |\widehat{v}_{\omega_{n-1}}|.$$

Prove that the j^{th} *singular value of* $\mathrm{circ}(\mathbf{v}) \in \mathbb{C}^{n \times n}$ *satisfies* $\sigma_j(\mathrm{circ}(\mathbf{v})) = \sqrt{n}\, |\widehat{v}_{\omega_j}|$.

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 (2.33) 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 2.4.3 and (2.29)). Hence, (2.33) can potentially help evaluate convolutional layers of neurons more quickly via (2.30). In addition, convolutions also appear in numerous other important applications, two of which we will briefly discuss next.

Example 2.4.12 (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 if we have both the blurry signal $\mathbf{u} \star \mathbf{v} \in \mathbb{C}^n$ *and 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* $\mathrm{circ}(\mathbf{v})\mathbf{u} = \mathbf{u} \star \mathbf{v}$ *for* $\mathbf{u} \in \mathbb{C}^n$*. In such a case we can instead always find an approximate solution by returning, e.g., the least-squares estimate* $\tilde{\mathbf{u}} = \mathrm{circ}(\mathbf{v})^\dagger (\mathbf{u} \star \mathbf{v}) = P_{\mathcal{C}(\mathrm{circ}(\mathbf{v})^*)} \mathbf{u}$ *(recall Section 2.3.2). Furthermore, an SVD of* $\mathrm{circ}(\mathbf{v})$*, and therefore* $\mathrm{circ}(\mathbf{v})^\dagger$*, can be constructed efficiently using Theorem 2.4.11.*

Example 2.4.13 (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*

$$\begin{aligned} t(x) \;&=\; (q_2 x^2 + q_1 x + q_0)(r_2 x^2 + r_1 x + r_0) \\ &=\; \underbrace{q_2 r_2}_{t_4} x^4 + \underbrace{(q_2 r_1 + q_1 r_2)}_{t_3} x^3 + \underbrace{(q_2 r_0 + q_1 r_1 + q_0 r_2)}_{t_2} x^2 + \underbrace{(q_1 r_0 + q_0 r_1)}_{t_1} x + \underbrace{q_0 r_0}_{t_0}. \end{aligned}$$

In vector form this corresponds to

$$\begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} q_0 & 0 & 0 & q_2 & q_1 \\ q_1 & q_0 & 0 & 0 & q_2 \\ q_2 & q_1 & q_0 & 0 & 0 \\ 0 & q_2 & q_1 & q_0 & 0 \\ 0 & 0 & q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix} = \operatorname{circ}\left(\begin{pmatrix} \mathbf{q} \\ \mathbf{0} \end{pmatrix}\right) \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ 0 \\ 0 \end{pmatrix} \star \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ 0 \\ 0 \end{pmatrix}.$$

Hence, if we can compute (I)DFTs quickly then we can also multiply polynomials quickly via (2.33).

Exercise 2.4.18. *Consider the "finite difference" matrix $D_2 \in \mathbb{R}^{n\times n}$ whose entries are given by*

$$(D_2)_{i,j} = \begin{cases} -2 & \text{if } i = j \\ 1 & \text{if } (i-j) \equiv 1 \bmod n \\ 1 & \text{if } (i-j) \equiv n-1 \bmod n \\ 0 & \text{otherwise} \end{cases}. \tag{2.34}$$

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

$$(E)_{i,j} = \begin{cases} 2\cos(2\pi j/n) - 2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \tag{2.35}$$

Exercise 2.4.19. *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 (2.33) 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".

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 $f : \mathbb{R} \to \mathbb{R}$ 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 $\mathrm{e}^{\mathrm{i}\theta} = \cos(\theta) + \mathrm{i}\sin(\theta)$, 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?[10] 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.[11] 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.

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

[10] We urge the interested reader to consult, e.g,. [34, Chapters 1 and 3] to learn why this sixth type of constant-cost operation is indeed not *too* fishy after all. . . .

[11] 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.

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.[12]

Example 2.4.14 (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.[13]

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 2.4.15 (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 $k \in [m]$.

We can now see that computing $A\mathbf{x}$ can always be done using $\mathcal{O}(mn)$ operations.

Exercise 2.4.20. *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 $\mathcal{O}(g)$?*

(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) $f(\epsilon,x,y) = \frac{0.2\ x}{\epsilon^2} + \log y$

[12] We urge the interested reader to consult, e.g, [34, 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., [42].

[13] 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.

(d) $f(\epsilon, x, y) = \frac{20\sqrt{x}\log(100+\ln(y))}{\sqrt{\epsilon}}$

Exercise 2.4.21. *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.*[14]

2.4.3 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 2.4.14 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 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 [12].[15]

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 [25]). 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, 45] allowed the FFT to also be utilized effectively for vector sizes whose prime factorizations contain larger primes. This section has primarily followed [14, 5, 45]. For more information on Fourier methods and algorithms we recommend that the interested reader consult the relevant chapters of [44], [34], [15], or [7]. For a fast FFT implementation we recommend FFTW [22]. 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

$$\widehat{u}_\omega = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} u_j \, f_n^{\omega\cdot j}. \tag{2.36}$$

By splitting the sum (2.36) for $\widehat{u}_\omega$ into p_1 smaller subsums, one for each possible residue modulo p_1, we can see that

$$\widehat{u}_\omega = \frac{1}{\sqrt{n}} \sum_{k=0}^{p_1-1} f_n^{\omega\cdot k} \left(\sum_{j=0}^{\frac{n}{p_1}-1} u_{k+p_1\cdot j} \, f_n^{\omega\cdot p_1\cdot j} \right). \tag{2.37}$$

[14] 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],[50, 32, 53].

[15] The QR decomposition discussed in Section 2.2.4 also made the list of the top 10 most important algorithm developments by the way!

Let's now rewrite the internal sum of (2.37) 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,

$$\left(\mathbf{u}^{(k,p_1)}\right)_j := u_{k+j\cdot p_1} \tag{2.38}$$

for all $j \in [n/p_1]$.[16] Our equation (2.37) for $\widehat{u}_\omega$ now becomes

$$\widehat{u}_\omega = \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega\cdot k} \left(\frac{1}{\sqrt{n/p_1}} \sum_{j=0}^{\frac{n}{p_1}-1} \left(\mathbf{u}^{(k,p_1)}\right)_j f_n^{p_1\cdot\omega\cdot j} \right) \tag{2.39}$$

$$= \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega\cdot k} \left(\frac{1}{\sqrt{n/p_1}} \sum_{j=0}^{\frac{n}{p_1}-1} \left(\mathbf{u}^{(k,p_1)}\right)_j f_{\frac{n}{p_1}}^{\left(\omega \bmod \frac{n}{p_1}\right)\cdot j} \right)$$

$$= \frac{1}{\sqrt{p_1}} \sum_{k=0}^{p_1-1} f_n^{\omega\cdot k} \left(\widehat{\mathbf{u}^{(k,p_1)}}\right)_{\omega \bmod \frac{n}{p_1}}. \tag{2.40}$$

For the sake of clarity we emphasize that the vector $\widehat{\mathbf{u}^{(k,p_1)}} \in \mathbb{C}^{n/p_1}$ in (2.40) 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 (2.39) – (2.40) for yourself before reading further.

At this point it's useful to ask ourselves what we've managed to accomplish by reformulating (2.36) into (2.40). 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 $\widehat{\mathbf{u}^{(0,p_1)}}, \ldots, \widehat{\mathbf{u}^{(p_1-1,p_1)}}$ computed in the first step in order to compute each entry of $\widehat{\mathbf{u}}$ via (2.40). 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 (2.40) 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 (2.31) using $\sim n^2$ operations.

Although the computational gain obtained from (2.40) 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 13. Analogous sum-splitting leads to the **Inverse Fast Fourier Transform (IFFT)** which

[16] 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 13 FAST FOURIER TRANSFORM (FFT)

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** ω 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 $\widehat{\mathbf{u}}$

can be obtained from Algorithm 13 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 13 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 13 require $p_1 \cdot T_{\frac{n}{p_1}}$ operations while 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 13 is recursively invoked again to compute $\widehat{\mathbf{u}^{(0,p_1)}}, \ldots, \widehat{\mathbf{u}^{(p_1-1,p_1)}}$ 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 13's lines 3 – 5) we have

$$T_n = \mathcal{O}\left(n \cdot \sum_{\ell=1}^{m} p_\ell\right) + \mathcal{O}(n) = \mathcal{O}(m \cdot p_m \cdot n). \tag{2.41}$$

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 2.4.16. *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 2.4.16 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 13 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 (2.31) 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 13 with a $\mathcal{O}(n^2)$ runtime which, in practice, is even slower than the direct method (2.31).

The inability of Algorithm 13 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 2.4.22 (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.*

2.4.4 The FFT for Vectors of Arbitrary Size

As discussed in the previous subsection, Algorithm 13 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 13 which will be efficient for vectors of dimension $\tilde{n}$.

Let $\omega \in [n]$. We may rewrite $\widehat{u}_\omega$ as

$$\widehat{u}_\omega = f_n^{\frac{\omega^2}{2}} f_n^{-\frac{\omega^2}{2}} \cdot \widehat{u}_\omega = \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot \sum_{j=0}^{n-1} u_j \, f_n^{\omega \cdot j - \frac{\omega^2}{2}} = \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot \sum_{j=0}^{n-1} u_j \, f_n^{\frac{j^2}{2}} f_n^{\frac{-(\omega-j)^2}{2}} \tag{2.42}$$

Note that the last sum in (2.42) 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 (2.42) now reveals that

$$\widehat{u}_\omega = \frac{f_n^{\frac{\omega^2}{2}}}{\sqrt{n}} \cdot (\tilde{\mathbf{u}} \star \mathbf{v})_\omega \quad \forall \omega \in [n]. \tag{2.43}$$

Note that the convolution in (2.43) can be computed efficiently by the FFT and IFFT using (2.33) since $\tilde{n}$ is a power of two. Furthermore, $\tilde{n} \leq 4n$ by definition. Hence, we have established the following theorem.

Theorem 2.4.17. *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 2.4.23. *Finish the proof of Theorem 2.4.17 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* (2.42) – (2.43)*?*

Theorem 2.4.17 generalizes Theorem 2.4.16 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.

2.4.5 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 14 FAST TOEPLITZ MATRIX MULTIPLICATION

1: **Input:** $\mathbf{a} \in \mathbb{C}^{2n-1}$, $\mathbf{v} \in \mathbb{C}^n$
2: **Output:** $\text{Toep}_n(\mathbf{a})\mathbf{v} \in \mathbb{C}^n$
3: $\mathbf{c} \leftarrow (a_{n-1}, a_{n-2}, \ldots, a_0, 0, a_{2n-2}, \ldots, a_n)^T \in \mathbb{C}^{2n}$
4: Compute $\widehat{\mathbf{c}} \in \mathbb{C}^{2n}$ using the FFT
5: Compute $\widehat{\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}} \in \mathbb{C}^{2n}$ using the FFT
6: $\widehat{\mathbf{b}} \leftarrow \sqrt{2n}\, \widehat{\mathbf{c}} \odot \widehat{\begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}} \in \mathbb{C}^{2n}$
7: Compute $\mathbf{b} \in \mathbb{C}^{2n}$ using the IFFT
8: Return $(b_0, b_1, \ldots, b_{n-1})^T \in \mathbb{C}^n$

Recalling Lemma 2.4.7 we can begin by embedding $\text{Toep}_n(\mathbf{a})$ into a $2n \times 2n$ circulant matrix. Specifically, we have seen that the vector $\mathbf{c} = (a_{n-1}, a_{n-2}, \ldots, a_0, 0, a_{2n-2}, \ldots, a_n)^T \in \mathbb{C}^{2n}$ satisfies $(\text{circ}(\mathbf{c}))_{j,k} = (\text{Toep}_n)(\mathbf{a})_{j,k}$ for all $j, k \in [n]$. This then further implies that $(\text{Toep}_n(\mathbf{a})\mathbf{v})_j = \left(\text{circ}(\mathbf{c}) \begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}\right)_j$ for all $j \in [n]$ by (2.30). Hence, we can compute $\text{Toep}_n(\mathbf{a})\mathbf{v}$ by computing the convolution $\mathbf{c} \star \begin{pmatrix}\mathbf{v}\\ \mathbf{0}\end{pmatrix}$. Finally, we further seen in (2.33) 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 14 for streamlined pseudocode.

Considering the runtime of Algorithm 14, 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 2.4.17. Hence, Algorithm 14 will always utilize a total of $\mathcal{O}(n \log n)$ operations in order to compute $\text{Toep}_n(\mathbf{a})\mathbf{v} \in \mathbb{C}^n$. 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 14 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, $\text{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m \times n}$, 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 $\text{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m\times n}$ 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 14.[17]

Example 2.4.18. *Let $\mathbf{a} \in \mathbb{C}^6$ and $\mathbf{v} \in \mathbb{C}^2$. Suppose that we want to compute $\text{Toep}_{5,2}(\mathbf{a})\mathbf{v} \in \mathbb{C}^5$. Instead of computing the result directly we can instead, e.g., decompose $\text{Toep}_{5,2}(\mathbf{a})$ into two 2×2 and two 1×1 Toeplitz submatrices, and then compute $\text{Toep}_{5,2}(\mathbf{a})\mathbf{v}$ using the resulting block-matrix form via*

$$\text{Toep}_{5,2}(\mathbf{a})\mathbf{v} = \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \\ a_2 & a_3 \\ a_1 & a_2 \\ a_0 & a_1 \end{pmatrix}\mathbf{v} = \begin{pmatrix} \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \end{pmatrix} \\ \\ \begin{pmatrix} a_2 & a_3 \\ a_1 & a_2 \end{pmatrix} \\ \\ \begin{pmatrix} a_0 \end{pmatrix} \quad \begin{pmatrix} a_1 \end{pmatrix} \end{pmatrix}\mathbf{v} = \begin{pmatrix} \begin{pmatrix} a_4 & a_5 \\ a_3 & a_4 \end{pmatrix}\mathbf{v} \\ \\ \begin{pmatrix} a_2 & a_3 \\ a_1 & a_2 \end{pmatrix}\mathbf{v} \\ \\ \begin{pmatrix} a_0 \end{pmatrix} v_0 + \begin{pmatrix} a_1 \end{pmatrix} v_1 \end{pmatrix}.$$

Note that the fact that $\text{Toep}_{5,2}(\mathbf{a})$ has constant diagonals ensures that all of its square submatrices above are also Toeplitz.

Exercise 2.4.24. *Suppose that we are given $\text{Toep}_{m,n}(\mathbf{a}) \in \mathbb{C}^{m\times n}$ and integers $1 \leq p \leq \min\{m,n\}$, $h \in [m-p+1]$, and $\ell \in [n-p+1]$. Let $A \in \mathbb{C}^{p\times p}$ be such that $A_{j,k} := \big(\text{Toep}_{m,n}(\mathbf{a})\big)_{h+j,\ell+k}$ for all $j,k \in [p]$. Show that A is Toeplitz.*

Exercise 2.4.25. *Let $p,n \in \mathbb{N}$, $\text{Toep}_{pn,n}(\mathbf{a}) \in \mathbb{C}^{pn\times n}$, and $\mathbf{v} \in \mathbb{C}^n$. Show that $\text{Toep}_{pn,n}(\mathbf{a})\mathbf{v} \in \mathbb{C}^{pn}$ can be computed in $\mathcal{O}(pn\log n)$ operations.*

Exercise 2.4.26. *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 2.4.27. *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*

$$g(x) = \sum_{\omega\in\mathbb{Z}} c_\omega \mathrm{e}^{2\pi\mathbb{i}\omega x} \quad \forall x \in [0,1],$$

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

[17] 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*

$$(F\boldsymbol{u})_j = \sqrt{n} \sum_{\omega \equiv j \bmod n} c_\omega.$$

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 2.4.3) also have other computational advantages. Consider, e.g., a convolutional layer of neurons $\ell : \mathbb{R}^N \to \mathbb{R}^N$ defined by $\ell(\mathbf{x}) := \sigma(W\mathbf{x} + \mathbf{b})$ 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 $\sigma(W\mathbf{x} + \mathbf{b})$ 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 14. 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!

IGNORE – STUFF FOR POSSIBLE(?) FUTURE INCLUSION:
(1) adding in a subsection about symmetric positive definite/semi-definite matrices....

– Example in complexity section showing that $n/\epsilon + m/\delta + 100n + 2$ is $\mathcal{O}\left(\frac{\max\{m,n\}}{\min\{\epsilon,\delta\}}\right)$.

– Trace and it's properties

– possitive (semi) def matrices and equivalent defs

Definition 2.4.19 (Frobenius Norm)**.** *The Froebenius norm of* $A \in \mathbb{C}^{n\times N}$ *is*

$$\|A\|_F = \sqrt{\sum_{\ell,j} |A_{\ell,j}|^2} = \sqrt{Trace(A^*A)}$$

$$\begin{aligned}
\|A\|_F &= \sqrt{\text{Trace}(A^*A)} \\
&= \sqrt{\text{Trace}(V\Sigma^*U^*U\Sigma V^*)} \\
&= \sqrt{\text{Trace}(V\Sigma^*\Sigma V^*)} \\
&= \sqrt{\text{Trace}(V^*V\Sigma^*\Sigma)} \\
&= \sqrt{\text{Trace}(\Sigma^*\Sigma)} \\
&= \sqrt{\sum_{j=1}^{\min(n,N)} (\sigma_j(A))^2}
\end{aligned}$$

where we have used the cyclic property of the trace. Thus the $\|A\|_F$ is equivalent to the ℓ^2-norm of a vector formed by the singular values of A

– Trace and some properties of trace, Frob. norm innner product

– ℓ^p-norms and Holder inequality

ADD BELOW INTO SVD SECION

DATA FITTING:

- Given $P = \{\vec{x}_1, \cdots, \vec{x}_N\} \subseteq \mathbb{R}^D$.

- Our fitness measure for an affine subspace H is $R_\tau(H,P) = (\sum_{\vec{x}_j \in P} d(\vec{x}_j, H)^\tau)^{1/\tau}$ for some $\tau \in \mathbb{R}^+$. Here $d(\cdot,\cdot)$ is Hausdorff distance.

- Assume that P has mean $\frac{1}{N}\sum_{j=1}^{N} \vec{x}_j = \vec{0}$

- For $\tau = 2$ we get a least squares approximation to P.

- Review $\tau = 2$: This can be solved exactly in $O(ND\min\{N,D\})$-time

Goal : Minimize $(R_2(H,P))^2 = \sum_{\vec{x}\in P}^{N} d(\vec{x}_j, H)^2$ over all $n < D$ dimensional subspace H.

- Let $X_P = (\vec{x}_1, \cdots, \vec{x}_N) \in \mathbb{R}^{D\times N}$

- Represent an n-dimensional H(subspace) by a projection matrix $\Pi_H \in \mathbb{R}^{D\times D}$(rank n) that projects onto H.

$$\begin{aligned}
(R_2(H,P))^2 &= \sum_{\vec{x}_j \in P} d(\vec{x}_j, H)^2 \\
&= \sum_{\vec{x}_j \in P} \|\vec{x}_j - \Pi_H \vec{x}_j\|_2^2 \\
&= \|X_P - \Pi_H X_P\|_F^2 \ (\textit{Recall } \|A\|_F^2 = \sum\sum |a_{ij}|^2) \\
&= \|(I - \Pi_H) X_P\|_F^2
\end{aligned}$$

- We want to minimize this $\|\cdot\|_F$ over all H. Recall that

$$\begin{aligned}\|A\|_F &= \sqrt{trace(A^T A)} \\ &= \sqrt{trace(V\Sigma^2 V^T))} \\ &= \sqrt{trace(\Sigma^2)}\ (when\ A = U\Sigma V^T, the\ SVD\ of A) \\ &= \sqrt{\sum_{j=1}^{N} \sigma_j(A)^2}\end{aligned}$$

So we want to minimize $\sum_{j=1}^{min(N,D)} \sigma_j((I - \Pi_H)X_P)^2$ over all H.

- If H is n-dimensional, $(I - \Pi_H)$ is $(D - n)$-dimensional projection.
- Let $X_P = U\Sigma V^T$ (SVD of X_P).
- We should let $I - \Pi_H$ project onto the subspace spanned by $D - n$ columns of U associated with $\sigma_D, \cdots, \sigma_{n+1}$.

To minimize $R_2(P, H)$ over H, we want to

1. calculate SVD of X_P, $X_P = U\Sigma V^T$.
2. set $\Pi_H = U_n U_n^T$ where $U_n = \begin{pmatrix}\vec{u}_1 & \cdots & \vec{u}_n & \vec{0} & \cdots \vec{0}\end{pmatrix}$; $U = \begin{pmatrix}\vec{u}_1 & \cdots & \vec{u}_D\end{pmatrix}$.

Chapter 3

A Review of Introductory Probability with Applications to Big Data

Probability is fundamental to large scale data analysis and algorithm development for a least two reasons. First, some problems are simply too time consuming to solve exactly in general, even when potential solutions are relatively simple to check for correctness (e.g., so-called NP-hard problems such as the Traveling Salesman Problem fall into this category). For this type of extremely hard problem one can often "guess" a near-best solution, however, with the help of a little randomness. Though an extremely important and interesting area of study, such randomized approximation techniques are not going to be the focus of our applications in this chapter.

In this chapter we instead focus on the "Big Data Regime" where even trivial computations are too difficult to carry out exactly due to the scale of the data involved. This has certainly become an increasingly common scenario over the past several decades as the proliferation of cell phones, laptops, and intelligent devices of all kinds communicating with one another across the internet with increasing rapidity have resulted in an exploding torrent of data – too much to even store in many cases. In this setting the problems one seeks to solve are often extremely easy conceptually, but still practically impossible to solve exactly due to the shear size of the data involved. For example, one might "simply" want to compute the average of 10^{1000} numbers – an utterly trivial task if it weren't for the fact that even looking at each of the numbers once to sum them up would take more than a lifetime. In such situations probability can come to the rescue by, e.g., guiding approaches for subsampling the data in a way that preserves the quantities (e.g., the average) that you want to approximate.

More recently, the availability of huge amounts of training data has resulted in the machine learning and artificial intelligence revolution. In this brave new world probabilistic

methods are used to not only help collect, process, and store the initial training data from which learning occurs, but also to, e.g., help train neural networks based on the stored training data after its been harvested. As a result, it is truly impossible to understand how modern machine learning methods work without having a solid grasp on probability.

Now equipped with the motivation to do the work of understanding some probability, we will begin with a review of the principal actors and definitions. As we do, exercises will appear below. To learn the material best we strongly encourage you to do the exercises using only this reference, your brain, a pen/pencil/piece of chalk, and a piece of paper or chalkboard. Using anything electronic to help you is a demerit. How few demerits can you get while completing all the exercises? The fewer, the better your understanding will be. That said, looking up answers and supplementary information is sometimes necessary, and totally OK – but resist it at all costs until you have been stuck for awhile. Real understanding results from confusion and struggle – if the chapter's exercises seem too easy, it's likely because you are either not really learning anything new, or because you are looking up too many solutions (or both). If the chapter seems too hard, find the first exercise you can't do, review all the definitions involved in its statement, and seek additional references/assistance to help you understand those definitions as needed. Some recommended probability references are provided in Section 3.9.

3.1 Probability Densities and Random Vectors

We will begin by defining random vectors (and real numbers). Both are entirely determined by a given *probability density* herein. Probability densities, in turn, will always be defined in one of three ways below: They will either be determined by a *probability density function*, a finite sum of *Dirac detlas*, or a weighted sum of both. Each of these options is reviewed next.

Definition 3.1.1. *A* **Probability Density Function (PDF)** *is a non-negative*[1] *function* $p : \mathbb{R}^n \to [0, \infty)$ *for which*

$$\int_{\mathbb{R}^n} p(\mathbf{x}) \; d\mathbf{x} \; = \; 1.$$

A **random vector** $X \in \mathbb{R}^n$ **with probability density function** p *represents a value in* $\mathbb{R}^n$*. It takes on a particular value in any given set* $\mathcal{S} \subseteq \mathbb{R}^n$ *with probability*

$$\mathbb{P}\left[X \in \mathcal{S}\right] := \mathbb{P}_X\left[\mathcal{S}\right] := \int_{\mathcal{S}} p(\mathbf{x}) \; d\mathbf{x} \in [0, 1].^2$$

[1] We will also generally assume that PDFs are always (at least) piecewise continuous unless otherwise noted.

[2] Here $\mathbb{P}_X$ is an example of a probability measure. We only work with Lebesgue measurable sets $\mathcal{S} \subseteq \mathbb{R}^n$ and functions p for which such integrals are well defined throughout the book. We do not, however, assume the reader has had a course in measure theory – as a result, if this footnote doesn't make sense to you, don't worry! It can be very safely ignored.

Finally, if $n = 1$ then X may also be called a **random variable**, *or a* **random number**.

We will now present some fundamental examples of random vectors defined by PDFs. The most important of these are *normal (or Gaussian) random vectors.*

Example 3.1.2 (**Uniform Random Vectors on $(a,b)^n$**). *A uniform random vector $X \in \mathbb{R}^n$ on $(a,b)^n$ has the probability density function*

$$p(\mathbf{x}) = \begin{cases} \frac{1}{(b-a)^n} & \text{if } \mathbf{x} \in [a,b]^n \\ 0 & \text{otherwise} \end{cases}$$

for given real numbers $a < b$. A uniform random vector $X \in \mathbb{R}^n$ on $(a,b)^n$ is denoted by $X \sim \text{Unif}((a,b)^n)$.

Example 3.1.3 (**Normal Random Numbers**). *A normal random variable $X \in \mathbb{R}$ has the probability density function*

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)^2} \tag{3.1}$$

for a choice of parameters $\mu \in \mathbb{R}$ and $\sigma \in (0,\infty)$. A normal random number is denoted by $X \sim N(\mu,\sigma)$, and also sometimes referred to as a Gaussian random variable.

Example 3.1.4 (**Standard Normal Random Vector with Mean $\boldsymbol{\mu} \in \mathbb{R}^n$**). *A standard normal random vector $X \in \mathbb{R}^n$ with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ has the probability density function*

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}} e^{-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2}}. \tag{3.2}$$

A standard normal random vector with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ is denoted by $X \sim N(\boldsymbol{\mu}, I_n)$.

Exercise 3.1.1. *Check that* (3.2) *is indeed a probability density function.*
Hint: We suggest that you first recall/show that (3.1) *is PDF. Having established that fact it should then be much easier to show that* (3.2) *is also a PDF.*

One of the easiest ways to obtain new random variables is by taking functions of random variables one already has access to. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ and suppose that $X \in \mathbb{R}^n$ is a random vector with PDF p_X. Then $f(X) \in \mathbb{R}^m$ will also be a random vector satisfying

$$\mathbb{P}_{f(X)}[\mathcal{S}] = \mathbb{P}[f(X) \in \mathcal{S}] = \mathbb{P}\left[X \in f^{-1}(\mathcal{S})\right] = \mathbb{P}_X\left[f^{-1}(\mathcal{S})\right] = \int_{f^{-1}(\mathcal{S})} p_X(\mathbf{x})\, d\mathbf{x}, \tag{3.3}$$

where $f^{-1}(\mathcal{S}) := \{\mathbf{y} \in \mathbb{R}^n \mid f(\mathbf{y}) \in \mathcal{S}\} \subseteq \mathbb{R}^n$ is the set of inputs that f maps into $\mathcal{S} \subseteq \mathbb{R}^m$. The following examples illustrate how this observation can be used productively in practice.

Example 3.1.5 (**Standard Folded Normal**). *The standard folded normal random number is $|X|$ where $X \sim N(0,1)$. Its PDF can be found in the following way for any interval $(a,b) \subset [0,\infty)$. We have that*

$$\begin{aligned}\mathbb{P}_{|X|}\left[(a,b)\right] = \mathbb{P}_X\left[(a,b)\cup(-b,-a)\right] &= \frac{1}{\sqrt{2\pi}}\int_a^b \mathrm{e}^{-\frac{x^2}{2}}\,dx + \frac{1}{\sqrt{2\pi}}\int_{-b}^{-a}\mathrm{e}^{-\frac{x^2}{2}}\,dx \\ &= \frac{2}{\sqrt{2\pi}}\int_a^b \mathrm{e}^{-\frac{x^2}{2}}\,dx = \sqrt{\frac{2}{\pi}}\int_a^b \mathrm{e}^{-\frac{x^2}{2}}\,dx.\end{aligned}$$

The result of this calculation combined with the fact that $f(x) = |x|$ maps $\mathbb{R}$ onto the nonnegative reals is that we may now assert that a standard folded normal random number admits the PDF

$$p(x) = \begin{cases}\sqrt{\frac{2}{\pi}}\mathrm{e}^{-\frac{x^2}{2}} & \text{if } x > 0 \\ 0 & \text{otherwise}\end{cases}.$$

Example 3.1.6 (**The Box-Muller Transform [6]**). *The Box-Muller transform is method for transforming uniformly distributed random vectors $X \in \mathbb{R}^2$ into standard normal random vectors. This transform is particularly useful when one wants to work with Gaussian random variables in a low-level programming language that only offers access to uniformly random number generators. Let $X \sim \mathrm{Unif}((0,1)^2)$ and consider the function $f : (0,1)^2 \to \mathbb{R}^2$ defined by $(u,v) = f\left((x,y)\right) = \left(\sqrt{-2\ln(x)}\cos(2\pi y),\ \sqrt{-2\ln(x)}\ \sin(2\pi y)\right)$. We claim that $f(X) \sim N(\mathbf{0}, I_2)$. To show this, we begin by noting that f is an invertible function with*

$$(x,y) = f^{-1}((u,v)) = \left(\mathrm{e}^{-(u^2+v^2)/2},\ \frac{1}{2\pi}\begin{cases}\cot^{-1}(u/v) & \text{if } v > 0 \\ \pi + \cot^{-1}(u/v) & \text{otherwise}\end{cases}\right).$$

As a result, after changing variables[3] *we have that for any $(a,b)\times(c,d) \subset \mathbb{R}^2$*

$$\begin{aligned}\mathbb{P}_{f(X)}\left[(a,b)\times(c,d)\right] = \mathbb{P}_X\left[f^{-1}\left((a,b)\times(c,d)\right)\right] &= \int_{f^{-1}((a,b)\times(c,d))} dx\ dy \\ &= \int_a^b\int_c^d \left|\frac{\partial(x,y)}{\partial(u,v)}\right| du\ dv \\ &= \int_a^b\int_c^d \left|\left(-u\mathrm{e}^{-(u^2+v^2)/2}\right)\left(\frac{1}{2\pi}\frac{u}{v^2+u^2}\right) - \left(-v\mathrm{e}^{-(u^2+v^2)/2}\right)\left(\frac{1}{2\pi}\frac{-v}{v^2+u^2}\right)\right| du\ dv \\ &= \frac{1}{2\pi}\int_a^b\int_c^d \mathrm{e}^{-(u^2+v^2)/2}\ du\ dv.\end{aligned}$$

Looking back at (3.2) we can now see that $f(X)$ will indeed be a standard mean $\mathbf{0}$ normal random vector $\in \mathbb{R}^2$ as claimed.

[3] For an introductory review of the change of variables theorem and its related notation we suggest looking at, e.g., [38, Section 6.2].

Exercise 3.1.2. *A chi-squared random number is X^2 where $X \sim N(0,1)$. Show that a chi-squared random number admits the PDF* $p(x) = \begin{cases} \sqrt{\frac{1}{2\pi x}}\mathbb{e}^{-x/2} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$.

Exercise 3.1.3. *Suppose that $X \sim N(\mu, \sigma)$ for fixed constants $\mu \in \mathbb{R}$ and $\sigma \in (0, \infty)$.*

(a) Show that the random number $(X - \mu) \sim N(0, \sigma)$.

(b) Show that the random number $\frac{X}{\sigma} \sim N(\mu/\sigma, 1)$.

(c) Show that the random number $\frac{X-\mu}{\sigma} \sim N(0, 1)$.

One practical weakness of random vectors $X \in \mathbb{R}^n$ that admit PDFs in our setting is that they will generally never take on any particular fixed value in $\mathbb{R}^n$ (or even any element from a fixed finite set of values)! This is somewhat awkward in the setting of digital computers where every random variable must, in practice, ultimately take on one of only a finite set of possible floating point values. As a result, we are interested in making sure that the probability framework we use in this book allows us to easily incorporate discrete random variables which always do take on one of a predictable set of finite values.

To be more precise, suppose that $X \in \mathbb{R}^n$ is a random vector with a PDF $p : \mathbb{R}^n \to [0, \infty)$, and fix a particular $\mathbf{c} \in \mathbb{R}^n$. The probability that X actually takes on the value $\mathbf{c}$ is

$$\mathbb{P}[X = \mathbf{c}] \;:=\; \lim_{\epsilon \to 0^+} \mathbb{P}_X[B(\mathbf{c}, \epsilon)] \;=\; \lim_{\epsilon \to 0^+} \int_{B(\mathbf{c},\epsilon)} p(\mathbf{x})\, d\mathbf{x} \;\geq\; 0$$

where $B(\mathbf{c}, \epsilon) := \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{c}\|_2 < \epsilon\} \subset \mathbb{R}^n$ denotes the open Euclidean (i.e., ℓ^2-norm) ball of radius $\epsilon > 0$ centered at $\mathbf{c} \in \mathbb{R}^n$. If p is bounded on a closed ball of any positive radius around $\mathbf{c}$ (so that there exists a constant $\gamma > 0$ with $\sup_{\mathbf{x} \in \overline{B(\mathbf{c},\gamma)}} p(\mathbf{x}) \leq M \in (0, \infty)$), we can immediately see that

$$\mathbb{P}[X = \mathbf{c}] \;=\; \lim_{\epsilon \to 0^+} \int_{B(\mathbf{c},\epsilon)} p(\mathbf{x})\, d\mathbf{x} \;\leq\; M \lim_{\epsilon \to 0^+} \int_{B(\mathbf{c},\epsilon)} d\mathbf{x} \;=\; M \lim_{\epsilon \to 0^+} \text{Vol}(B(\mathbf{c}, \epsilon)) = 0.$$

As a result, random variables with bounded PDFs (which are all that we will consider herein) will take on any value $\mathbf{c}$ you might be interested in with probability zero. To not have this be the case, we will need a different kind of probability density.

3.1.1 Discrete Random Variables and Dirac Densities

The following notation will allow us to easily treat both continuous and discrete random variables together within the same framework. This extremely useful notation will be motivated by the notion of a dirac delta function, $\delta : \mathbb{R}^n \to \{0, \infty\}$, intuitively "defined" as follows

$$\delta(\mathbf{x}) = \begin{cases} \infty & \text{if } \mathbf{x} = \mathbf{0} \\ 0 & \text{if } \mathbf{x} \neq \mathbf{0} \end{cases}, \text{ satisfying } \int_{\mathbb{R}^n} \delta(\mathbf{x})\, d\mathbf{x} = 1.$$

Given its eventual utility, let's take a moment now to perform some thought experiments about what other properties our strange δ function should have given its "definition" above. We urge you to stop and participate in the following experiments as they are performed, as our next important definition can seem quite odd without the intuition they provide.

First, since δ is zero everywhere except at the origin, it should be the case that $\int_{\mathcal{S}} \delta(\mathbf{x})\ d\mathbf{x} = 0$ should hold for all $\mathcal{S} \subset \mathbb{R}^n$ not containing $\mathbf{0}$, correct? Given this, it also seems reasonable that we should have $\int_{\mathcal{S}} \delta(\mathbf{x})\ d\mathbf{x} = 1$ hold for all $\mathcal{S} \subset \mathbb{R}^n$ that do contain $\mathbf{0}$ (since the integral over the complement of $\mathcal{S}$ will contribute nothing to the integral of δ over all of $\mathbb{R}^n$ in this case). As such, we seem inevitably lead to the conclusion that we will also have

$$\int_{\mathcal{S}} \delta(\mathbf{x})\ d\mathbf{x} \ = \ \begin{cases} 1 & \text{if } \mathbf{0} \in \mathcal{S} \\ 0 & \text{if } \mathbf{0} \notin \mathcal{S} \end{cases}.$$

be true for all $\mathcal{S} \subset \mathbb{R}^n$. Furthermore, if $\delta : \mathbb{R}^n \to \{0, \infty\}$ can be treated like any other function, it seems reasonable that we should be able to, e.g., shift it by any constant we want, $\mathbf{c} \in \mathbb{R}^n$, and still have the regular rules of integration tell us that $\int_{\mathbb{R}^n} \delta(\mathbf{x} - \mathbf{c})\ d\mathbf{x} = 1$ holds. This shifted function $\delta(\mathbf{x} - \mathbf{c})$ is now zero everywhere except at $\mathbf{x} = \mathbf{c}$, moreover, so it seems natural to therefore assert that we should definitely also have

$$\int_{\mathcal{S}} \delta(\mathbf{x} - \mathbf{c})\ d\mathbf{x} \ = \ \begin{cases} 1 & \text{if } \mathbf{c} \in \mathcal{S} \\ 0 & \text{if } \mathbf{c} \notin \mathcal{S} \end{cases}$$

hold even when $\mathbf{c} \neq \mathbf{0}$. If you haven't seen Dirac delta notation before, we strongly urge you to stop here and carefully think this last paragraph through!

Continuing our thought experiment concerning our rather strange δ function, we can further assert that linearity of integration should also mean that, e.g.,

$$\int_{\mathcal{S}} 2\delta(\mathbf{x})\ d\mathbf{x} = 2\int_{\mathcal{S}} \delta(\mathbf{x})\ d\mathbf{x} = \begin{cases} 2 & \text{if } \mathbf{0} \in \mathcal{S} \\ 0 & \text{if } \mathbf{0} \notin \mathcal{S} \end{cases}.$$

Moreover, the fact that 2 is a constant here doesn't seem to matter much after recalling that δ itself is only nonzero at the origin! Really, any function that is 2 at the origin should "look like" the constant function 2 to our very strange δ function, right? Taking this intuitive thought to its natural limit, we conclude that perhaps it makes sense for

$$\int_{\mathcal{S}} f(\mathbf{x})\delta(\mathbf{x})\ d\mathbf{x} = \begin{cases} f(\mathbf{0}) & \text{if } \mathbf{0} \in \mathcal{S} \\ 0 & \text{if } \mathbf{0} \notin \mathcal{S} \end{cases}$$

to hold for any "reasonable" function $f : \mathbb{R}^n \to \mathbb{R}$. And here, perhaps you will agree, we should maybe stop our experiment given that it seems to be getting increasingly fanciful. Nonetheless, one can in fact formalize these thought experiments with some care, and then

use them to good effect.[4] Taking this fact as given, we will define the following notation which can be safely and consistently used for calculations throughout the remainder of this text.

Definition 3.1.7. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be continuous, $\mathcal{S} \subseteq \mathbb{R}^n$, and $\mathbf{c} \in \mathbb{R}^n$.* **Dirac delta**, *$\delta$, notation will be used herein to evaluate formal integrals according to the rule*

$$\delta_{\mathbf{c},\mathcal{S}}[f] := \int_{\mathcal{S}} f(\mathbf{x})\delta(\mathbf{x}-\mathbf{c})\ d\mathbf{x} \ := \ \begin{cases} f(\mathbf{c}) & \text{if } \mathbf{c} \in \mathcal{S} \\ 0 & \text{if } \mathbf{c} \notin \mathcal{S} \end{cases}.$$

Exercise 3.1.4. *The formal Dirac delta notation defined above is motivated by the notion of a generalized function, or distribution.[5] This exercise will help you understand a little better one way that δ can be defined as an object that one can "integrate functions against" using limits. Let $\chi_j : \mathbb{R}^n \to \mathbb{R}$ be defined by*

$$\chi_j(\mathbf{x}) := \begin{cases} j^n & \text{if } \mathbf{x} \in \left(\frac{-1}{2j}, \frac{1}{2j}\right)^n \\ 0 & \text{otherwise} \end{cases}$$

for all $j \in \mathbb{N}$. Choose any continuous $f : \mathbb{R}^n \to \mathbb{R}^m$, closed $\mathcal{S} \subset \mathbb{R}^n$, and $\mathbf{c} \in \mathbb{R}^n$ you like. Show that

(a) $\lim_{j\to\infty} \int_{\mathcal{S}} f(\mathbf{x})\chi_j(\mathbf{x}-\mathbf{c})\ d\mathbf{x} = 0$ *if $\mathbf{c} \in \mathcal{S}^c$, and that*

(b) $\lim_{j\to\infty} \int_{\mathcal{S}} f(\mathbf{x})\chi_j(\mathbf{x}-\mathbf{c})\ d\mathbf{x} = f(\mathbf{c})$ *if $\mathbf{c} \in$ the interior of $\mathcal{S}$.*

As a consequence of parts (a) and (b) above, conclude that $\delta_{\mathbf{c},\mathcal{S}}[f]$ in Definition 3.1.7 is given by $\lim_{j\to\infty} \int_{\mathcal{S}} f(\mathbf{x})\chi_j(\mathbf{x}-\mathbf{c})\ d\mathbf{x}$ for all continuous $f : \mathbb{R}^n \to \mathbb{R}^m$ and closed $\mathcal{S} \subset \mathbb{R}^n$ as long as $\mathbf{c} \notin$ the boundary of S. Finally, to remove the restriction that $\mathbf{c} \notin$ the boundary of $\mathcal{S}$, argue that one may, e.g., represent $\delta_{\mathbf{c},\mathcal{S}}[f]$ by the double limit

(c) $\lim_{\epsilon\to 0} \left(\lim_{j\to\infty} \int_{\mathcal{S}+B(\mathbf{0},\epsilon)} f(\mathbf{x})\chi_j(\mathbf{x}-\mathbf{c})\ d\mathbf{x} \right)$

for any continuous $f : \mathbb{R}^n \to \mathbb{R}^m$, closed $\mathcal{S} \subset \mathbb{R}^n$, and $\mathbf{c} \in \mathbb{R}^n$. Here we note that $\mathcal{S} + B(\mathbf{0},\epsilon) := \{\mathbf{x}+\mathbf{y} \mid \mathbf{x} \in \mathcal{S} \text{ and } \mathbf{y} \in B(\mathbf{0},\epsilon)\} \subset \mathbb{R}^n$ is an open set $\forall \epsilon > 0$.

[4] One certainly needs to be careful, however. The main issue with our informal "definition" of δ was, of course, that it is "∞ at the origin". The "size of ∞" was then somewhat constrained by the integral condition $\int_{\mathbb{R}^n} \delta(\mathbf{x})\ d\mathbf{x} = 1$. But, ∞ is a tricky concept that needs extreme care when it's used. For example, ideas about what, e.g., δ^2 (or other functions composed with δ) might be are best left alone, especially herein, and most likely don't make any real sense at all. In fact, δ itself only really makes sense inside of an integral – any other time you see one, it's really just there to indicate a rule about how integrals should be evaluated. To get a better idea of how integrals involving δ can be defined as limits of integrals involving well defined functions, we direct you to the optional Exercise 3.1.4.

[5] If you are interested in learning more about the mathematical background of classical Dirac deltas and distributions, we recommend reading, e.g., [19, Chapter 9]. For our purposes herein, however, the formal definition provided by Definition 3.1.7 will suffice.

Using Dirac notation we can now also define discrete random variables in terms of their Dirac densities.

Definition 3.1.8. *A* **discrete random variable** $X \in \mathbb{R}^n$ *is determined by a finite set of discrete probabilities* $\{p_1, \ldots, p_N\} \subset (0, 1]$ *satisfying* $\sum_{j=1}^N p_j = 1$*, and always takes on a value in a finite set* $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^n$*. More generally, it will take on some value in a given set* $\mathcal{S} \subseteq \mathbb{R}^n$ *with probability*

$$\mathbb{P}\left[X \in \mathcal{S}\right] := \mathbb{P}_X\left[\mathcal{S}\right] := \int_{\mathcal{S}} \sum_{j=1}^N p_j \delta(\mathbf{x} - \mathbf{c}_j)\, d\mathbf{x} \;=\; \sum_{j=1}^N p_j \int_{\mathcal{S}} \delta(\mathbf{x} - \mathbf{c}_j)\, d\mathbf{x} \in [0, 1].^6$$

Finally, if $n = 1$ *then* X *be may also be called a* **discrete random number**.

From above we see that the probability density of a discrete random vector $X \in \mathbb{R}^n$ which takes on a set of values $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^n$ each with probability $\{p_1, \ldots, p_N\} \subset (0, 1]$, respectively, can formally be denoted by the sum $p(x) = \sum_{j=1}^N p_j \delta(\mathbf{x} - \mathbf{c}_j)$. Choosing one of the vectors $\mathbf{c}_j$ from above, we can further see that

$$\mathbb{P}\left[X = \mathbf{c}_j\right] \;=\; \lim_{\epsilon \to 0^+} \;\mathbb{P}_X\left[B(\mathbf{c}_j, \epsilon)\right] \;=\; \lim_{\epsilon \to 0^+} \int_{B(\mathbf{c}_j, \epsilon)} \sum_{j=1}^N p_j \delta(\mathbf{x} - \mathbf{c}_j)\, d\mathbf{x} \;=\; p_j \;>\; 0.$$

As a result, it is also common to write the probability density of a discrete random variable over a set of values $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^n$ as

$$p(\mathbf{x}) = \sum_{j=1}^N \mathbb{P}\left[X = \mathbf{c}_j\right] \delta(\mathbf{x} - \mathbf{c}_j).$$

Example 3.1.9 (A Fair Coin Flip Model)**.** *Let* X *be the value of a fair coin flip where a heads represents a value of one, and a tails represents zero. The probability density of* X *is given by*

$$p(x) \;=\; \mathbb{P}\left[X = 0\right] \delta(x - 0) + \mathbb{P}\left[X = 1\right] \delta(x - 1) \;=\; \frac{1}{2}\delta(x) + \frac{1}{2}\delta(x - 1).$$

Example 3.1.10 (Rademacher Random Variables)**.** *We will say that* $X \in \mathbb{R}$ *is a Rademacher random number if its probability density is given by*

$$p(x) \;=\; \frac{1}{2}\delta(x - 1) + \frac{1}{2}\delta(x + 1).$$

That is, it takes on the values ± 1 *each with probability* $1/2$*.*

[6] In this framework each $p_j \in \mathbb{R}$ (as well as the number 1 when δ is not being explicitly multiplied by anything) is interpreted as a constant function.

Exercise 3.1.5. *Write the probability density for the result of a fair six-sided die roll.*

Exercise 3.1.6. *Let $X \in \mathbb{R}^n$ be a discrete random vector that takes on values in $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\}$. Show that $\mathbb{P}_X[\mathcal{S}] = \mathbb{P}_X[\mathcal{S} \cap \{\mathbf{c}_1, \ldots, \mathbf{c}_N\}] = \sum_{\mathbf{c}_j \in \mathcal{S}} \mathbb{P}[X = \mathbf{c}_j]$ holds for all $\mathcal{S} \subseteq \mathbb{R}^n$, where empty sums are defined to be 0.*

As a couple final remarks about Diracs, it is occasionally convenient to use a Dirac as a way to consider a fixed constant $\mathbf{c} \in \mathbb{R}^n$ to be a "random" quantity with probability density $p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{c})$. A bit more generally, we will say that random vector $X \in \mathbb{R}^n$ is uniformly distributed on a nonempty finite set $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^n$ if it's a discrete random variable which takes each value in $\mathcal{C}$ with probability $\frac{1}{|\mathcal{C}|}$. Note that if $\mathcal{C}$ is a singleton set, we are again effectively considering a non-random constant that takes on the value $\mathbf{c}_1$ with probability 1.

3.1.2 General Densities

Going forward, we will consider the probability density of every random vector $X \in \mathbb{R}^n$ discussed hereafter to always be of the form

$$p_X(\mathbf{x}) = \lambda p(\mathbf{x}) + (1 - \lambda) \sum_{j=1}^{N} p_j \delta(\mathbf{x} - \mathbf{c}_j) \tag{3.4}$$

where $\lambda \in [0, 1]$, $p : \mathbb{R}^n \to [0, \infty)$ is a PDF, $N \in \mathbb{Z}^+$, $\{p_j\}_{j=1}^N \subset (0, 1]$ with $\sum_j p_j = 1$, and $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^n$. As above, we then define $\mathbb{P}_X[\mathcal{S}] := \int_{\mathcal{S}} p_X(\mathbf{x})\, d\mathbf{x}$ for all $\mathcal{S} \subset \mathbb{R}^n$. Finally, we also note that we always consider $\mathbb{P}_X[\emptyset] = \int_{\emptyset} p_X(\mathbf{x})\, d\mathbf{x} := 0$ for all densities p_X.

Exercise 3.1.7. *Verify that any probability density p_X as defined in (3.4) always satisfies $\int_{\mathbb{R}^n} p_X(\mathbf{x})\, d\mathbf{x} = 1$.*

Example 3.1.11 (A Sparse Random Variable)**.** *Suppose we want the density of a random number $X \in \mathbb{R}$ which is 0 with probability 1/2, and distributed like a Gaussian with probability 1/2. Its density is*

$$p(x) = \frac{1}{2}\delta(x) + \frac{1}{2\sigma\sqrt{2\pi}} e^{-\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)^2}.$$

Note that we expect such a random number to be 0 half the time, and therefore a sequence of its realizations should be somewhat sparse. In contrast, a standard Gaussian random variable will never be exactly 0 (i.e., a standard normal will be 0 with probability 0).

Exercise 3.1.8. *Write down a probability density for a random number that is distributed like a Rademacher with probability $q \in (0, 1)$, and distributed like a normal random variable with probability $1 - q$.*

We are now in the position to verify our first general probability inequality. Though simple, it is probably the most commonly used probability inequality in existence.

Theorem 3.1.12 (Union Bound)**.** *Let $N \in \mathbb{Z}^+$, $\mathcal{S}_j \subseteq \mathbb{R}^n$ for $j \in [N]$, and $X \in \mathbb{R}^n$ be a random variable with probability density p. Then,*

$$\mathbb{P}_X\left[\bigcup_{j\in[N]} \mathcal{S}_j\right] \leq \sum_{j\in[N]} \mathbb{P}_X[\mathcal{S}_j].$$

Furthermore, equality holds when the S_j are all mutually disjoint.

Proof. We proceed by induction. The base case is trivially true since $\mathbb{P}[\mathcal{S}_0] \leq \mathbb{P}[\mathcal{S}_0]$. The induction hypothesis is thus that

$$\mathbb{P}_X\left[\bigcup_{j=0}^{k-1} S_j\right] \leq \sum_{j=0}^{k-1} \mathbb{P}_X[S_j]$$

holds. Note that for any two sets $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$, $\mathbb{P}[\mathcal{A} \cup \mathcal{B}] = \mathbb{P}[\mathcal{A}] + \mathbb{P}[\mathcal{B}] - \mathbb{P}[\mathcal{A} \cap \mathcal{B}]$ (see Exercise 3.1.9). Thus,

$$\mathbb{P}_X\left[\bigcup_{j=0}^{k} \mathcal{S}_j\right] = \mathbb{P}_X\left[\bigcup_{j=0}^{k-1} \mathcal{S}_j\right] + \mathbb{P}_X\left[\mathcal{S}_k\right] - \mathbb{P}_X\left[\mathcal{S}_k \cap \left(\bigcup_{j=0}^{k-1} \mathcal{S}_j\right)\right].$$

Since all probabilities are non-negative, we have by our induction hypothesis that

$$\mathbb{P}_X\left[\bigcup_{j=0}^{k} \mathcal{S}_j\right] \leq \mathbb{P}_X\left[\bigcup_{j=0}^{k-1} \mathcal{S}_j\right] + \mathbb{P}_X\left[\mathcal{S}_k\right] \leq \sum_{j=0}^{k-1} \mathbb{P}_X[\mathcal{S}_j] + \mathbb{P}_X[\mathcal{S}_k] = \sum_{j=0}^{k} \mathbb{P}_X[\mathcal{S}_j]$$

also holds. Furthermore, we can also see that equality will hold if $\mathcal{S}_k \cap \left(\bigcup_{j=0}^{k-1} \mathcal{S}_j\right) = \emptyset$ holds for all $k \in [N] \setminus \{0\}$. □

Exercise 3.1.9. *Let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$. Verify that any random vector $X \in \mathbb{R}^n$ with a probability density p_X as defined in (3.4) will satisfy $\mathbb{P}_X[\mathcal{A} \cup \mathcal{B}] = \mathbb{P}_X[\mathcal{A}] + \mathbb{P}_X[\mathcal{B}] - \mathbb{P}_X[\mathcal{A} \cap \mathcal{B}]$.*[7]

[7] If this exercise seems a little "mushy" to you it's likely because you have good mathematical instincts! In fact, a better way to formalize the proof of exercises like this is to use measure theory. If the mush bothers you, we urge you to use it as a motivation to learn about Lebesgue measure/integration (we suggest, e.g., the latest edition of Royden's classic book [46] in that case). In the meantime, you can simply convince yourself that something extremely odd would have to be going on for this to not hold and then except it as an axiom.

3.2 Expectations, Moments, and Some Related Inequalities

In many applications the probability density of a random vector $X \in \mathbb{R}^n$ of interest is simply too complicated to be of much practical value, if one is even lucky enough to know it at all. In such situations one instead often attempts to learn something about the probability density of X by instead (approximately) computing some of its so-called "moments". We shall see several examples of this later in the chapter. For the time being, however, it will benefit to us to first define what moments are, as well as to see what they can tell us about the probability that a given random variable will take on certain values. We will begin by first defining the expectation of a random vector.

Definition 3.2.1 (Expectation)**.** *Let $X \in \mathbb{R}^n$ be a random vector with probability density p_X. The* **expectation***, or* **mean***, or* **expected value** *of a function $f : \mathbb{R}^n \to \mathbb{R}^q$ of X, $f(X) \in \mathbb{R}^q$, is the quantity*

$$\mathbb{E}[f(X)] := \int_{\mathbb{R}^n} f(\mathbf{y})\, p_X(\mathbf{y})\, d\mathbf{y} \ \in \mathbb{R}^q.^8$$

When the function above is the identity, $f(\mathbf{x}) = \mathbf{x}$, the resulting quantity $\mathbb{E}[X] = \int_{\mathbb{R}^n} \mathbf{y}\, p_X(\mathbf{y})\, d\mathbf{y} \in \mathbb{R}^n$ is simply called the **expectation of** *X.*

Exercise 3.2.1. *Carefully write out the following expectation calculations to demonstrate your understanding of Definition 3.2.1.*[9]

(a) Let $X \sim \text{Unif}((a,b))$ be a uniform random variable on $(a,b) \subset \mathbb{R}$ (recall Example 3.1.2). Show that $\mathbb{E}[X] = (a+b)/2$.

(b) Let $X \sim \text{Unif}((a,b)^n)$ be a uniform random vector on $(a,b)^n \subset \mathbb{R}^n$. Show that $\mathbb{E}[X] = ((a+b)/2, (a+b)/2, \ldots, (a+b)/2) \in \mathbb{R}^n$.

(c) Let $X \sim N(\mu, \sigma)$ be a normal random variable (recall Example 3.1.3). Show that $\mathbb{E}[X] = \mu$.

(d) Let $X \in \mathbb{R}^n$ be a standard normal random vector $X \in \mathbb{R}^n$ with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ (recall Example 3.1.4). Show that $\mathbb{E}[X] = \boldsymbol{\mu}$.

(e) Let X be the value of a fair coin flip as in Example 3.1.9. Compute $\mathbb{E}[X]$.

(f) Let $X \in \mathbb{R}$ is a Rademacher random number (recall Example 3.1.10). Compute $\mathbb{E}[X]$.

(g) Let $X \in \mathbb{R}$ be defined as in Example 3.1.11. Compute $\mathbb{E}[X]$.

[8] Note that in general $\mathbb{E}[f(X)]$ is the integral of a vector-valued function. Recall that if $f(\mathbf{y}) = (f_1(\mathbf{y}), \ldots, f_q(\mathbf{y})) \in \mathbb{R}^q$, then $\int_{\mathbb{R}^n} f(\mathbf{y})\, p_X(\mathbf{y})\, d\mathbf{y} := \left(\int_{\mathbb{R}^n} f_1(\mathbf{y})\, p_X(\mathbf{y})\, d\mathbf{y}, \ldots, \int_{\mathbb{R}^n} f_q(\mathbf{y})\, p_X(\mathbf{y})\, d\mathbf{y}\right) \in \mathbb{R}^q$.

[9] This is a fundamental definition: we strongly recommend that you carefully do every part of this exercise to cement your understanding!

Exercise 3.2.2. *Let $X \in [0, \infty)$ be a non-negative random variable. Show that $\mathbb{E}[X] = \int_0^\infty \mathbb{P}[X > t]\ dt$.*

HINT: Consider selectively using that $y = \int_0^\infty \mathbb{1}_{[0,y)}(t)\ dt$, where $\mathbb{1}_{[0,y)}(t) = \begin{cases} 1 \text{ if } 0 \leq t < y \\ 0 \text{ else} \end{cases}$,

when computing $\mathbb{E}[X] = \int_0^\infty y\ p_X(y)\ dy$.

Having now equipped ourselves with the expectation of a function of a random variable, we are now able to define moments, absolute moments, and L^p-norms of random variables.

Definition 3.2.2 (Moments, Absolute Moments, and L^p-Norms of Random Variables)**.** *Let $p \in [1, \infty)$, and $X \in \mathbb{R}$ be a random variable with probability density p_X. Then,*

- *The p^{th} **moment of** X is $\mathbb{E}[X^p] = \int_{\mathbb{R}} y^p\ p_X(y)\ dy$.*
- *The p^{th} **absolute moment of** X is $\mathbb{E}[|X|^p] = \int_{\mathbb{R}} |y|^p\ p_X(y)\ dy$.*
- *The L^p **norm of** X is $\|X\|_{L^p} := (\mathbb{E}[|X|^p])^{1/p} = \sqrt[p]{\int_{\mathbb{R}} |y|^p\ p_X(y)\ dy}$.*

Exercise 3.2.3. *Let X be a random variable and $p \in [1, \infty)$. Use the result of Excercise 3.2.2 together with a change of variable to show that*

$$\mathbb{E}[|X|^p] = \int_0^\infty p t^{p-1} \mathbb{P}[|X| > t]\ dt.$$

Can $\mathbb{E}[|X|^p]$ be finite if there exist constants $c, c' > 0$ such that $\mathbb{P}[|X| > t] \geq ct^{-p}\ \forall t \geq c'$?

The following fundamental inequality will allow us to use (absolute) moments of an arbitrary random variable X in order to learn things about its potentially unknown probability measure $\mathbb{P}_X$. This will be of critical importance in later applications where we initially know nothing about a random variable's probability density or measure beyond being able to estimate several of its moments. What can X's moments tell use about $\mathbb{P}_X$ in such cases? The mighty Markov inequality will soon begin to answer this question for us.

Theorem 3.2.3 (Markov's inequality)**.** *Let $f : \mathbb{R}^n \to [0, \infty)$ be continuous, $a > 0$, and $X \in \mathbb{R}^n$ be a random vector. Then,*

$$\mathbb{P}_{f(X)}\left[[a, \infty]\right] = \mathbb{P}\left[f(X) \geq a\right] \leq \frac{\mathbb{E}\left[f(X)\right]}{a}.^{10}$$

Proof. As usual, let p_X denote the probability density of X. We have that

$$\mathbb{E}\left[f(X)\right] = \int_{\mathbb{R}^n} f(\mathbf{y})\ p_X(\mathbf{y})\ d\mathbf{y} \geq \int_{\underbrace{\{\mathbf{y} | f(\mathbf{y}) \geq a\}}_{f^{-1}([a,\infty)) \subseteq \mathbb{R}^n}} f(\mathbf{y})\ p(\mathbf{y})\ d\mathbf{y}$$

[10] Note that we have left open the possibility that f grows so quickly that $\mathbb{E}\left[f(X)\right]$ is undefined (i.e., infinite). In that case we consider this statement to be vacuously true.

since f is non-negative valued. Furthermore, by the definition of $f^{-1}([a,\infty))$ in combination with (3.3) we can continue to see that

$$\begin{aligned}\mathbb{E}[f(X)] &\geq \int_{f^{-1}([a,\infty))} f(\mathbf{y})\ p_X(\mathbf{y})\ d\mathbf{y}\ \geq\ \int_{f^{-1}([a,\infty))} a\ p_X(\mathbf{y})\ d\mathbf{y} \\ &= a\int_{f^{-1}([a,\infty))} p_X(\mathbf{y})d\mathbf{y} = a\ \mathbb{P}_{f(X)}[[a,\infty)]\,.\end{aligned}$$

Dividing the inequality above through by $a > 0$ now yields the desired result. □

Using the Markov inequality we can quantify the following claim: A positive random variable is unlikely to ever be more than a few times larger than its expectation. To see why, consider a positive random number $X \in (0,\infty)$, and a function that does not change the value of X, $f(X) = |X|$. Now choose $b > 0$ and apply Theorem 3.2.3 with the constant $a = b\ \mathbb{E}[X]$. Doing so, we obtain that

$$\mathbb{P}[X \geq b\ \mathbb{E}[X]] \leq \frac{\cancel{\mathbb{E}[X]}}{b\ \cancel{\mathbb{E}[X]}} = \frac{1}{b}.$$

Setting, e.g., $b = 3$ now tells us that *any* nonnegative random variable X will be less than 3 times its expectation with probability at least two thirds. As a result, if we can simply estimate the expectation of X we can already bound the probability that X is too large from above.

Exercise 3.2.4. *Let $X \sim \text{Unif}((0,12))$. Use Markov's inequality to bound the probability $\mathbb{P}[X \geq 8]$ from above. Then, compute $\mathbb{P}[|X| \geq 8]$ exactly. How loose (or tight) is the upper bound provided by Markov's inequality in this case?*

With the Markov inequality at our disposal we will now be able to prove many additional results that bound the behavior of $\mathbb{P}_X$ in terms of moment information about X. In particular, we will make frequent use of the following Chebyshev inequality (Theorem 3.2.5) which utilizes additional moment information about a random variable in the form of its so-called "variance".

Definition 3.2.4 (Variance)**.** *Let $X \in \mathbb{R}$ be a random variable. Then, the* **variance of** X *is* $\text{Var}[X] := \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$.

Note that since $\mathbb{E}[X]$ is always just a constant, we have for any random number $X \in \mathbb{R}$ that

$$\begin{aligned}0\ \leq\ \text{Var}[X] &= \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \int_{\mathbb{R}} (y - \mathbb{E}[X])^2\ p_X(y)\ dy \\ &= \int_{\mathbb{R}} y^2\ p_X(y)\ dy - \int_{\mathbb{R}} 2y\mathbb{E}[X]\ p_X(y)\ dy + \int_{\mathbb{R}} (\mathbb{E}[X])^2\ p_X(y)\ dy \qquad (3.5) \\ &= \mathbb{E}[X^2] - 2(\mathbb{E}[X])^2 + (\mathbb{E}[X])^2\ =\ \mathbb{E}[X^2] - (\mathbb{E}[X])^2\,.\end{aligned}$$

As a consequence, we can see that knowing both the expectation and the variance of X is equivalent to knowing both the first and second moments of X.

Exercise 3.2.5. *Let $a \in \mathbb{R}$ and X be a random variable. Show that* $\text{Var}[aX] = a^2 \text{ Var}[X]$.

Exercise 3.2.6. *Compute the variance of each random variable below.*

(a) Let $X \sim \text{Unif}((a,b))$ be a uniform random variable on $(a,b) \subset \mathbb{R}$ (recall Example 3.1.2). Compute $\text{Var}[X]$.

(b) Let $X \sim N(\mu, \sigma)$ be a normal random variable (recall Example 3.1.3). Show that $\text{Var}[X] = \sigma^2$.

(c) Let X be the value of a fair coin flip as in Example 3.1.9. Compute $\text{Var}[X]$.

(d) Let $X \in \mathbb{R}$ is a Rademacher random number (recall Example 3.1.10). Compute $\text{Var}[X]$.

(e) Let $X \in \mathbb{R}$ be defined as in Example 3.1.11. Compute $\text{Var}[X]$.

Theorem 3.2.5 (Chebyshev's Inequality)**.** *Let $X \in \mathbb{R}$ be a random variable with finite $\mu := \mathbb{E}[X]$ and $\sigma := \sqrt{\text{Var}[X]} > 0$. Then for all $k > 0$,*

$$\mathbb{P}[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$

Proof. We will apply the Markov inequality to the positive random variable $f(X) = |X - \mu|^2$ with $a = k^2\sigma^2$. Doing so we obtain that

$$\begin{aligned}\mathbb{P}[|X - \mu| \geq k\sigma] &= P\left[|X - \mu|^2 \geq k^2\sigma^2\right] \leq \frac{\mathbb{E}\left[|X - \mu|^2\right]}{k^2\sigma^2} \\ &= \frac{\mathbb{E}\left[(X - \mu)^2\right]}{k^2\sigma^2} = \frac{\text{Var}[X]}{k^2\sigma^2} = \frac{1}{k^2}.\end{aligned}$$

□

In statistical discussions one occasionally hears surprised statements like "Wow! That's four standard deviations above the mean!". With the help of Chebyshev's inequality we can begin to understand why a random variable X being multiple standard deviations above its mean is uncommon enough to be surprising. First, it's helpful to know that the **standard deviation of** X is defined to be $\sqrt{\text{Var}[X]}$ (i.e., that quantity σ in Theorem 3.2.5 above). Equipped with this knowledge, we can now see that Chebyshev's inequality tells us that the probability of X being more than 4 times its standard deviation from its mean μ is at most 1/16 (i.e., at least 15 of 16 realizations of any random variable will be within 4 standard deviations of it's mean μ). Thus, observing a realization of a random variable

being four standard deviations above the mean is indeed pretty surprising – we'd expect it to happen at most $1/16^{\text{th}}$ of the time (probably even less in fact, given that Chebyshev's inequality is not tight in general).

Exercise 3.2.7. *Let $X \sim N(0,1)$. Use Chebyshev's inequality to bound the probability $\mathbb{P}[|X| \geq 2]$ from above. Then, use your favorite mathematics software to compute $\mathbb{P}[|X| \geq 2]$ exactly (or, more correctly, up to machine precision). How loose (or tight) is the upper bound provided by Chebyshev's inequality in this case?*

As a final observation, it is interesting to note that (3.5) further shows that $\mathbb{E}[X^2] \geq (\mathbb{E}[X])^2$ holds for all random variables X. In fact, this is a special case of a much more general result that can be proven using Jensen's inequality.

Theorem 3.2.6 (Jensen's Inequality)**.** *Let X be a random variable, and $f : \mathbb{R} \to \mathbb{R}$ be a continuous convex function (e.g., where f'' exists at all but a finite number of points, and satisfies $f''(x) \geq 0$ for all x where it exists).*[11] *Then,*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

Exercise 3.2.8. *Use Jensen's inequality to prove that $\|X\|_{L^p} \leq \|X\|_{L^q}$ holds for all $1 \leq p \leq q < \infty$.*

Exercise 3.2.9 (Challenge Problem)**.** *Use Jensen's inequality together with the fact that $\|X+Y\|_{L^p} \leq \|X\|_{L^p} + \|Y\|_{L^p}$ holds for all random variables X, Y and values $p \in [1,\infty)$ (known as Minkowski's inequality) to prove that $\|X - \mathbb{E}[X]\|_{L^p} \leq 2\|X\|_{L^p}$ holds for all $p \in [1,\infty)$.*

Note the challenge problem just above includes an inequality that involves two different random variables at the same time. This is the first time we have encountered such a situation in this chapter, and it merits study for many reasons. In particular, in later sections we will repeat random computations multiple times in the hope that multiple somewhat inaccurate random approximations of a quantity we care about can be combined to provide a much higher quality final approximation. In order to understand how well such aggregated random approximations of multiple random variables will work, however, we first need to understand how to properly think about collections of random variables. We will begin this process in our next section.

---

[11] Examples of such functions include $f(x) = |x|$ which has both $f'(x) = \text{sign}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$, and $f''(x) = 0$, for all $x \neq 0$.

3.3 Collections of Random Vectors: Joint Densities, Marginals, and Independence

Anytime we are interested in the probability that a collection of random vectors $X_1 \in \mathbb{R}^{n_1}, \cdots, X_m \in \mathbb{R}^{n_m}$ takes on a collective set of outcomes, we need to consider their joint probability density.

Definition 3.3.1 (Joint Probability Densities)**.** *Let $X_1 \in \mathbb{R}^{n_1}, \ldots, X_m \in \mathbb{R}^{n_m}$ be a collection of random vectors/variables. The* **joint probability density** *of $X_1 \in \mathbb{R}^{n_1}, \ldots, X_m \in \mathbb{R}^{n_m}$ is a probability density of the form* (3.4) *with*

$$p_{(X_1,\ldots,X_m)}(\mathbf{x}) = \lambda p(\mathbf{x}) + (1-\lambda)\sum_{j=1}^{N} p_j \delta(\mathbf{x} - \mathbf{c}_j)$$

where $\lambda \in [0,1]$, $p : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m} \to [0,\infty)$ is a PDF, $N \in \mathbb{Z}^+$, $\{p_j\}_{j=1}^N \subset (0,1]$ with $\sum_j p_j = 1$, and $\{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$. The probability that $(X_1, X_2, \cdots, X_m) \in \mathcal{S} \subseteq \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$ is then given by

$$\mathbb{P}_{(X_1,\ldots,X_m)}[\mathcal{S}] := \int_{\mathcal{S}} p_{(X_1,\ldots,X_m)}(\mathbf{y})\, d\mathbf{y} \in [0,1] \quad \forall \mathcal{S} \subseteq \mathbb{R}^{\sum_j n_j}.$$

Example 3.3.2 (The General Multivariate Gaussian, or Normal, Distribution)**.** *Let $\boldsymbol{\mu} \in \mathbb{R}^n$, and $\Sigma \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix. We say that a random vector $X = (X_1, \ldots, X_n)$ has a multivariable Gaussian, or normal, distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ, denoted by $X \sim N(\boldsymbol{\mu}, \Sigma)$, if its probability density is given by*

$$p_X(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

Here $|\Sigma|$ denotes the determinant of Σ. We further note that $p_X = p_{(X_1,\ldots,X_n)}$ is also an example of a joint probability density of its n random entries $X_1, \ldots, X_n \in \mathbb{R}$.

Note that we already effectively discussed joint probability densities in Section 3.1. In particular, as pointed out in Example 3.3.2 above, the probability density of a random vector $X \in \mathbb{R}^n$ as per (3.4) is nothing more than the joint probability density of its n individual random entries. (You should check this!) The concept of a marginal probability density is slightly more involved, however, requiring additional formalism concerning Dirac deltas (recall Definition 3.1.7). For any $\mathbf{x} \in \mathbb{R}^n$ we will allow ourselves to represent $\delta(\mathbf{x})$ as $\delta(\mathbf{x}) = \prod_{i=1}^n \delta(x_i)$.[12] Doing so we can see that if, e.g., $\mathbf{c} \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \subseteq \mathbb{R}^n$ we may use

[12] Note here that only Dirac deltas evaluated at different variables are multiplied by one another here. Products of the type $\delta(x)\delta(x)$ involving the same variable are nonsensical.

Fubini's theorem[13] to consistently write for any continuous $f : \mathbb{R}^n \to \mathbb{R}^m$ that

$$\int_{\mathcal{S}_1 \times \cdots \times \mathcal{S}_n} f(\mathbf{x})\delta(\mathbf{x} - \mathbf{c})\ d\mathbf{x} = \int_{\mathcal{S}_2 \times \cdots \times \mathcal{S}_n} f(c_1, x_2, \ldots, x_n) \prod_{i=2}^{n} \delta(x_i - c_i)\ dx_2 \cdots dx_n$$
$$= \cdots = \int_{\mathcal{S}_n} f(c_1, \ldots, c_{n-1}, x_n)\delta(x_n - c_n)\ dx_n = f(\mathbf{c})$$

as required. With this formalism in hand we may now define marginal probability densities.

Definition 3.3.3 (Marginal Probability Densities)**.** *The probability density of any individual $X_j \in \mathbb{R}^{n_j}$ is given by the* **marginal probability density of $p_{(X_1,\ldots,X_m)}$ for X_j** *defined as*

$$p_{X_j}(\mathbf{z}) := \int_{\mathbb{R}^{n_1}} \cdots \int_{\mathbb{R}^{n_{j-1}}} \int_{\mathbb{R}^{n_{j+1}}} \cdots \int_{\mathbb{R}^{n_m}} p_{(X_1,\ldots,X_m)}(\mathbf{y}_1, \ldots, \mathbf{y}_{j-1}, \mathbf{z}, \mathbf{y}_{j+1}, \ldots, \mathbf{y}_m) \prod_{i \neq j} d\mathbf{y}_i.$$

In effect, one obtains p_{X_j} by integrating $p_{(X_1,\ldots,X_m)}$ over all but its j^{th} variable.

Exercise 3.3.1. *Suppose that $X \sim N(\boldsymbol{\mu}, \Sigma)$ where $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix with positive entries on its diagonal (recall Example 3.3.2). Show that the marginal probability density of the j^{th} entry of X, $X_j \in \mathbb{R}$, is that of a Gaussian random number $X_j \sim N(\boldsymbol{\mu}_j, \sqrt{\Sigma_{j,j}})$ (see Example 3.1.3).*

Exercise 3.3.2. *If $X \sim \text{Unif}((a,b)^n)$, show that $X_j \sim \text{Unif}((a,b))$.*

Exercise 3.3.3. *Show that a marginal probability density p_{X_1} defined as per Definition 3.3.3 with $j = 1$ is still a probability density of the form (3.4) (i.e., show that $p_{X_1}(\mathbf{x}) = \tilde{\lambda}\tilde{p}(\mathbf{x}) + (1 - \tilde{\lambda}) \sum_{j=1}^{\tilde{N}} \tilde{p}_j \delta(\mathbf{x} - \tilde{\mathbf{c}}_j)$ for some PDF $\tilde{p}$, $\tilde{\lambda} \in [0,1]$, etc.. How does each of $\tilde{p}$, $\tilde{\lambda}$, etc. depend on the p, λ, etc. that make up $p_{(X_1,\ldots,X_m)}$?).*

Exercise 3.3.4. *Once you understand Definition 3.3.3 well enough, we claim that it's not difficult to see that you can use the same idea to define joint marginal densities for any subset you like of m random vectors $X_1, X_2, \cdots, X_m$ using their joint probability density $p_{(X_1,\ldots,X_m)}$. To test your understanding, use the idea behind Definition 3.3.3 to write down a formula for the joint marginal probability density $p_{(X_1,X_2)}$ of $X_1 \in \mathbb{R}^{n_1}$ and $X_2 \in \mathbb{R}^{n_2}$ in terms of the joint probability density $p_{(X_1,\ldots,X_4)}$ of $X_1 \in \mathbb{R}^{n_1}, \ldots, X_4 \in \mathbb{R}^{n_4}$.*

Now that we have the tools of joint and marginal probability distributions at our disposal, we can revisit our previous definition of expectation (see Definition 3.2.1). In particular, we have the following apparent generalization.

[13] You should consult, e.g., [38] for a refresher on Fubini's theorem if you don't remember what it says!

Definition 3.3.4 (The Expectation of Functions of Several Random Variables)**.** *Let $X_1 \in \mathbb{R}^{n_1}, \ldots, X_m \in \mathbb{R}^{n_m}$ be random vectors with joint probability density $p_{(X_1,\ldots,X_m)}$. The* **expectation***, or* **mean***, or* **expected value** *of a function $f : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m} \to \mathbb{R}^q$ of $X_1, \ldots, X_m$, $f(X_1, \ldots, X_m) \in \mathbb{R}^q$, is the quantity*

$$\mathbb{E}[f(X_1, \cdots, X_m)] = \int_{\mathbb{R}^{\sum_{j=1}^m n_j}} f(\mathbf{y}_1, \ldots, \mathbf{y}_m)\ p_{(X_1,\ldots,X_m)}(\mathbf{y}_1, \ldots, \mathbf{y}_m)\ d\mathbf{y}_1 \cdots d\mathbf{y}_m.$$

Exercise 3.3.5. *Let $f(X_1, \ldots, X_m) = X_j$ simply pick out the j^{th} random input vector $X_j \in \mathbb{R}^{n_j}$. Show that $\mathbb{E}[f(X_1, \cdots, X_m)] = \int_{\mathbb{R}^{n_j}} \mathbf{y}_j\ p_{X_j}(\mathbf{y}_j)\ d\mathbf{y}_j$, where p_{X_j} is the marginal probability density of $p_{(X_1,\ldots,X_m)}$ for X_j. As a consequence, we note that $\mathbb{E}[X_j] := \int_{\mathbb{R}^{n_j}} \mathbf{y}_j\ p_{X_j}(\mathbf{y}_j)\ d\mathbf{y}_j$ is defined with respect its marginal probability density p_{X_j} in the context of multiple random variables.*

Looking at Definition 3.3.4 we can see that it is effectively nothing but a slightly rephrased version of Definition 3.2.1! More specifically, we can see that the two definitions are indeed identical after substituting $X = (X_1, \ldots, X_m)$, $n = \sum_{j=1}^m n_j$, and $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_m)$ into Definition 3.2.1. In addition, these same substitutions also show that, e.g., Markov's Inequality (Theorem 3.2.3) still holds for functions f of several random variables $X_1, \ldots, X_m$ into $[0, \infty)$. (You should check this!) The critically thinking reader at this point should be asking themselves why we have bothered to write down Definition 3.3.4 at all if its just a reformulation of Definition 3.2.1. To help answer their question, we will now prove the most important theorem there is about the expectation of sums of random variables using this section's updated notation.

Theorem 3.3.5 (Linearity of Expectation)**.** *Let $\alpha_1, \alpha_2 \in \mathbb{R}$, and $X_1, X_2 \in \mathbb{R}^n$ be random vectors. Then,*

$$\mathbb{E}[\alpha_1 X_1 + \alpha_2 X_2] = \alpha_1 \mathbb{E}[X_1] + \alpha_2 \mathbb{E}[X_2].$$

Proof. Let $p_{(X_1,X_2)}$ be the joint probability density of X_1 and X_2. We have that

$$\begin{aligned}
\mathbb{E}[\alpha_1 X_1 + \alpha_2 X_2] &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} (\alpha_1 \mathbf{y}_1 + \alpha_2 \mathbf{y}_2) p_{(X_1,X_2)}(\mathbf{y}_1, \mathbf{y}_2)\ d\mathbf{y}_1 d\mathbf{y}_2 \\
&= \int_{\mathbb{R}^n} \alpha_1 \mathbf{y}_1 \left(\int_{\mathbb{R}^n} p_{(X_1,X_2)}(\mathbf{y}_1, \mathbf{y}_2)\ d\mathbf{y}_2 \right)\ d\mathbf{y}_1 \\
&\qquad + \int_{\mathbb{R}^n} \alpha_2 \mathbf{y}_2 \left(\int_{\mathbb{R}^n} p_{(X_1,X_2)}(\mathbf{y}_1, \mathbf{y}_2) d\mathbf{y}_1 \right) d\mathbf{y}_2 \\
&= \alpha_1 \int_{\mathbb{R}^n} \mathbf{y}_1\ p_{X_1}(\mathbf{y}_1)\ d\mathbf{y}_1 + \alpha_2 \int_{\mathbb{R}^n} \mathbf{y}_2\ p_{X_2}(\mathbf{y}_2)\ d\mathbf{y}_2 \\
&= \alpha_1 \mathbb{E}[X_1] + \alpha_2 \mathbb{E}[X_2].
\end{aligned}$$

Here we use Fubini's Theorem to change the order of integration so that we can recover the marginal densities of X_1 and X_2. □

Exercise 3.3.6. *Generalize the proof of Theorem 3.3.5 to show that* $\mathbb{E}[\sum_{j=1}^{m} \alpha_j X_j] = \sum_{j=1}^{m} \alpha_j \mathbb{E}[X_j]$ *also holds for all* $m > 2$.

As Theorem 3.3.5 and the exercise just above demonstrate, linearity of expectation will hold for any random vectors $X_1, \ldots, X_m \in \mathbb{R}^n$ we consider herein. Of course, one might also expect that random vectors with more restricted types of joint probability distributions could also host many more remarkable properties. The next subsection will explore exactly this phenomena for the special class of random vectors whose joint probability distributions arise from independence assumptions.

3.3.1 Independent Random Variables

We are now ready to define the remarkably popular notion of independence.

Definition 3.3.6 (Independence)**.** *Let* $X_1 \in \mathbb{R}^{n_1}, \ldots, X_m \in \mathbb{R}^{n_m}$ *be random vectors with joint probability density* $p_{(X_1,\ldots,X_m)}$*. Denote the marginal probability density of* $p_{(X_1,\ldots,X_m)}$ *for each* X_j *by* p_{X_j}*. We say that* $X_1, \ldots, X_m$ *are* **mutually independent**, *or just* **independent**, *if*

$$p_{(X_1,\ldots,X_m)}(\mathbf{y}) = p_{(X_1,\ldots,X_m)}(\mathbf{y}_1, \ldots, \mathbf{y}_m) = \prod_{j=1}^{m} p_{X_j}(\mathbf{y}_j) \quad \forall \mathbf{y} \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}.$$

Example 3.3.7 (Independent Coin Flips)**.** *Let* $X_1, X_2 \in \{0, 1\}$ *be two random variables with joint probability density*

$$\begin{aligned} p_{(X_1,X_2)}(\mathbf{y}) &= \frac{1}{4}\delta(\mathbf{y} - (0,0)) + \frac{1}{4}\delta(\mathbf{y} - (0,1)) + \frac{1}{4}\delta(\mathbf{y} - (1,0)) + \frac{1}{4}\delta(\mathbf{y} - (1,1)) \\ &= \frac{1}{4}\delta(y_1)\delta(y_2) + \frac{1}{4}\delta(y_1)\delta(y_2 - 1) + \frac{1}{4}\delta(y_1 - 1)\delta(y_2) + \frac{1}{4}\delta(y_1 - 1)\delta(y_2 - 1). \end{aligned}$$

Here (X_1, X_2) *represent the result of two coin flips (e.g., where a* 0 *represents "Heads", and a* 1 *represents "Tails"). The two marginal probability densities for* X_1 *and* X_2 *are then*

$$\begin{aligned} p_{X_1}(y_1) &= \int_{\mathbb{R}} p_{(X_1,X_2)}(y_1, y_2)\, dy_2 = \frac{1}{2}\delta(y_1) + \frac{1}{2}\delta(y_1 - 1), \text{ and} \\ p_{X_2}(y_2) &= \int_{\mathbb{R}} p_{(X_1,X_2)}(y_1, y_2)\, dy_1 = \frac{1}{2}\delta(y_2) + \frac{1}{2}\delta(y_2 - 1). \end{aligned}$$

We can now see that X_1 *and* X_2 *are independent since*

$$p_{(X_1,X_2)}(\mathbf{y}) = p_{X_1}(y_1)p_{X_2}(y_2).$$

Intuitively, these two random variables are independent because the outcome of one tells you nothing about the potential outcome of the other (i.e., X_2*'s potential outcomes don't depend on whether* X_1 *is* 0 *or* 1*. It behaves the same no matter what* X_1 *does.)*

Example 3.3.8 (Dependent Coin Flips)**.** *Let $X_1, X_2 \in \{0,1\}$ be two random variables with joint probability density*

$$\begin{aligned} p_{(X_1,X_2)}(\mathbf{y}) &= \frac{1}{3}\delta(\mathbf{y}-(0,0)) + \frac{1}{3}\delta(\mathbf{y}-(0,1)) + \frac{1}{3}\delta(\mathbf{y}-(1,0)) \\ &= \frac{1}{3}\delta(y_1)\delta(y_2) + \frac{1}{3}\delta(y_1)\delta(y_2-1) + \frac{1}{3}\delta(y_1-1)\delta(y_2) \end{aligned}$$

again represent two coin flips. The two marginal densities for X_1 and X_2 are then

$$\begin{aligned} p_{X_1}(y_1) &= \int_{\mathbb{R}} p_{(X_1,X_2)}(y_1,y_2)\ dy_2 = \frac{2}{3}\delta(y_1) + \frac{1}{3}\delta(y_1-1), \text{ and} \\ p_{X_2}(y_2) &= \int_{\mathbb{R}} p_{(X_1,X_2)}(y_1,y_2)\ dy_1 = \frac{2}{3}\delta(y_2) + \frac{1}{3}\delta(y_2-1). \end{aligned}$$

We can now see that X_1 and X_2 are not independent (i.e., dependent) since

$$\begin{aligned} p_{(X_1,X_2)}(\mathbf{y}) &\neq p_{X_1}(y_1)p_{X_2}(y_2) \\ &= \frac{4}{9}\delta(y_1)\delta(y_2) + \frac{2}{9}\delta(y_1)\delta(y_2-1) + \frac{2}{9}\delta(y_1-1)\delta(y_2) + \frac{1}{9}\delta(y_1-1)\delta(y_2-1). \end{aligned}$$

Intuitively, we can also see that these coin flips must depend on one another due to the fact they can't both be 1 at the same time (i.e., if one of these coins is 1 then we know the other one can't be – their possible outcomes depend on one another).

Exercise 3.3.7. *Suppose that $X \sim N(\boldsymbol{\mu}, \Sigma)$ where $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix with positive entries on its diagonal (recall Example 3.3.2 and Exercise 3.3.1). Show that the entries of X, X_j, are mutually independent.*

Exercise 3.3.8. *Let $X \sim \text{Unif}((a,b)^n)$. Show that the entries of X, X_j, are mutually independent (recall Exercise 3.3.2).*

Independence is commonly used to implicitly define the joint probability density of a collection of random vectors whose marginal distributions are given explicitly. For example, two independent random variables $X \in \mathbb{R}$ and $Y \in \mathbb{R}$ will often be discussed having only specified the densities of X and Y individually. In this case the joint probability density of X and Y together, $p_{(X,Y)}$, is given by

$$p_{(X,Y)}(x',y') = p_X(x')p_Y(y'),$$

where p_X is the density of X, and p_Y is the density of Y.

Example 3.3.9 (Joint Density of Independent Uniform Random Variables)**.** *Let $X \sim \text{Unif}((1,2))$ and $Y \sim \text{Unif}((-1,0))$ be independent uniform random numbers. Their joint probability density function is*

$$p_{(X,Y)}(x',y') = \begin{cases} 1 & \text{if } (x',y') \in (1,2)\times(-1,0) \\ 0 & \text{otherwise} \end{cases}.$$

Exercise 3.3.9. *Let $X_j \sim N(\boldsymbol{\mu_j}, \sigma_j)$ for $j \in \{1, \ldots, n\}$ be independent Gaussian random numbers. Show that their joint density is*

$$p_{(X_1,\ldots,X_n)}(y_1, \ldots, y_n) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{y}-\boldsymbol{\mu})\right),$$

where $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $\Sigma_{j,j} = \sigma_j^2$. As a consequence, conclude that $(X_1, \ldots, X_n) \sim N(\boldsymbol{\mu}, \Sigma)$.

Exercise 3.3.10. *Let $f : \mathbb{R} \to \mathcal{S} \subseteq \mathbb{R}$ be a continuous bijection, and suppose that $X \in \mathbb{R}$ and $Y \in \mathbb{R}$ are independent random variables. Show that $f(X)$ and $f(Y)$ are also independent random variables.*
HINT: It suffices to show that $\mathbb{P}[(f(X), f(Y)) \in (a,b)\times(c,d)] = \mathbb{P}[f(X) \in (a,b)]\cdot\mathbb{P}[f(Y) \in (c,d)]$ holds for all open intervals $(a,b) \subset \mathbb{R}$ and $(c,d) \subset \mathbb{R}$.

To make implicit joint density definitions even easier we will also often use the following terminology.

Definition 3.3.10 (Identically Distributed)**.** *Two random variables $X_1, X_2 \in \mathbb{R}^n$ are identically distributed if their corresponding densities p_1 and p_2 satisfy $p_1 = p_2$ (i.e., if they both have the same density).*

Finally, if two random variables are "independent and identically distributed" we will often abbreviate this phrase and instead say they are "i.i.d.".

Example 3.3.11 (Joint Density of Independent Fair Coin Tosses)**.** *Let $X_1, \ldots, X_n \in \mathbb{R}$ be n i.i.d. fair coin tosses. To write down their joint density, we begin by noting that each density is $p_{X_j}(y) = \frac{1}{2}\delta(y) + \frac{1}{2}\delta(y-1)$. Therefore, their joint probability density $p_{(X_1,\ldots,X_n)}$ is*

$$p_{(X_1,\ldots,X_n)}(y_1, \ldots, y_n) := \prod_{j=1}^{n} p_{X_j}(y_j) = \prod_{j=1}^{n} \frac{1}{2}\delta(y_j) + \frac{1}{2}\delta(y_j - 1) = \frac{1}{2^n}\sum_{\mathbf{c}\in\{0,1\}^n} \delta(\mathbf{y}-\mathbf{c}).$$

Exercise 3.3.11. *Compute the following expectations.*

(a) Let X and Y be the random variables in Example 3.3.9. Compute the expectation of their product $\mathbb{E}[XY]$.

(b) Let $X_1, \ldots, X_n \in \mathbb{R}$ be n i.i.d. fair coin tosses as in Example 3.3.11. Compute the expectation of their product $\mathbb{E}\left[\prod_{j=1}^n X_j\right]$.

Exercise 3.3.12. *Let $X, Y \in \mathbb{R}$ be independent random variables. Show that the expectation of their product $Z = XY$ always satisfies $\mathbb{E}[Z] = \mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.*

3.3.2 Chernoff Inequalities and Variance Reduction by Averaging

Now we turn to a key fact about random variables and controlling variance: averaging across several i.i.d. copies of a random variable decreases the variance. We make this precise in the following theorem.

Theorem 3.3.12. *Let $X_1, \cdots, X_n \in \mathbb{R}$ be independent random variables satisfying $\mathbb{E}[X_i] = \mu \ \forall i \in [n]$. Then, the random variable $\frac{1}{n}\sum_{j=1}^n X_j$ satisfies both*

$$\text{Var}\left[\frac{1}{n}\sum_{j=1}^n X_j\right] \leq \frac{\max_j \text{Var}\left[X_j\right]}{n} \qquad \textit{and} \qquad \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^n X_j\right] = \mu.$$

Proof. By Theorem 3.3.5 and Exercise 3.3.6 we always have that

$$\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^n X_j\right] = \frac{1}{n}\sum_{j=1}^n \mathbb{E}[X_j] = \mu.$$

Now let $\gamma^2 := \max \mathbb{E}[X_j^2]$. By (3.5) we have that

$$\begin{aligned}
\text{Var}\left[\frac{1}{n}\sum_{j=1}^n X_j\right] &= \mathbb{E}\left[\left(\frac{1}{n}\sum_{j=1}^n X_j\right)^2\right] - \mu^2 \\
&= \mathbb{E}\left[\frac{1}{n^2}\left(\sum_{j=1}^n X_j^2 + \sum_{\substack{j\neq k \\ j,k\in[n]}} X_j X_k\right)\right] - \mu^2 \\
&= \frac{1}{n^2}\sum_{j=1}^n \mathbb{E}[X_j^2] + \frac{1}{n^2}\sum_{\substack{j\neq k \\ j,k\in[n]}} \mathbb{E}[X_j]\mathbb{E}[X_k] - \mu^2 \quad \text{(Exercises 3.3.6 and 3.3.12)} \\
&\leq \frac{\gamma^2}{n} + \left(\frac{n^2-n}{n^2}\right)\mu^2 - \mu^2 \\
&= \frac{1}{n}[\gamma^2 - \mu^2] \\
&= \frac{\max_j \text{Var}\left[X_j\right]}{n}. \qquad \text{(By (3.5))}
\end{aligned}$$

□

With variance reduction at our disposal we can now prove one of the most fundamental results of probability.

Exercise 3.3.13 ("Weak" Law of Large Numbers)**.** *Choose $\epsilon > 0$ as small as you like. Use Chebychev's inequality to argue that for a sequence of random variables $X_1, X_2, \cdots \in \mathbb{R}$ with bounded variances and the same expectations $\mathbb{E}[X_j] = \mu$ we have*

$$\lim_{n\to\infty} \mathbb{P}\left[\left|\frac{1}{n}\sum_{j=1}^{n} X_j - \mu\right| < \epsilon\right] = 1.$$

In Exercise 3.3.13 you proved that the average of a very large number of independent random variables will be close to its expectation with extremely high probability. In our next theorem we will specialize this result to a particular kind of discrete random variable taking on only 1's and 0's (i.e., what we have been calling "coin flips" so far). As motivation, let's consider how many 'Tails" (i.e., 1's) we can safely bet on being able to flip with a fair coin in 100 independent tries. Doing a quick expectation calculation reveals that we should get "about 50 Tails". However, that's not much comfort if I am asked to bet my house that I can get more than 40 Tails in 100 fair coin flips. How sure can I be that I won't end up homeless? The following theorem will help me decide how comfortable I should be betting my house on being able to accomplish this task. Before we proceed to the theorem, however, we need to recall some useful inequalities from calculus.

Fact 3.3.13. *For all $x \in \mathbb{R}$,*

$$1 + x \leq e^x.$$

As a result we can see that for $\forall c, x \in \mathbb{R}$ with $x > 0$,

$$\left(1 + \frac{c}{x}\right) \leq e^{c/x} \implies \left(1 + \frac{c}{x}\right)^x \leq e^c.$$

Exercise 3.3.14. *Prove Fact 3.3.13.*

With this fact in hand, we can now bound the probability that a sum of independent coin flips is significantly smaller than its mean.

Theorem 3.3.14 (A One-Sided Chernoff Inequality [40, Theorem 4.1])**.** *Let $I_1, \ldots, I_n \in \mathbb{R}$ be independent discrete random variables with probability densities $p_j(x) = \lambda_j \delta(x-1) + (1-\lambda_j)\delta(x)$ where $\lambda_j \in (0,1)$. Let $Y = \sum_{j=1}^n I_j$ with $\mu = \mathbb{E}[Y] = \sum_{j=1}^n \lambda_j$. Then, for $w \in (0,1)$ we have*

$$\mathbb{P}\left[Y \leq (1-w)\mu\right] \leq \left[\frac{e^{-w}}{(1-w)^{(1-w)}}\right]^{\mu}$$

Proof. Suppose $t > 0$. We can see that

$$\mathbb{P}\left[Y \leq (1-w)\mu\right] = \mathbb{P}\left[-tY \geq -t(1-w)\mu\right] = \mathbb{P}\left[e^{-tY} \geq e^{-t(1-w)\mu}\right].$$

We may now apply the Markov inequality (Theorem 3.2.3) to the right hand side to see that

$$\mathbb{P}\left[Y \leq (1-w)\mu\right] \leq \frac{\mathbb{E}\left[\exp\left(-tY\right)\right]}{\exp\left(-t(1-w)\mu\right)}. \tag{3.6}$$

Note that since the random variables are independent, we can write the sum in the exponential function as a product. Doing so we obtain

$$\begin{aligned}
\mathbb{E}\left[\exp\left(-tY\right)\right] &= \mathbb{E}\left[\exp\left(-t\sum_{j=1}^{n} I_j\right)\right] = \prod_{j=1}^{n} \mathbb{E}\left[\exp\left(-tI_j\right)\right] && \text{(Exercises 3.3.10 and 3.3.12)} \\
&= \prod_{j=1}^{n} \int_{\mathbb{R}} \exp\left(-tx\right)\left[\lambda_j \delta(x-1) + (1-\lambda_j)\delta(x)\right] dx \\
&= \prod_{j=1}^{n} \left[\lambda_j \exp\left(-t\right) + (1-\lambda_j)\exp\left(-t(0)\right)\right] \\
&= \prod_{j=1}^{n} \left[1 + \lambda_j\left(e^{-t} - 1\right)\right] \\
&\leq \prod_{j=1}^{n} e^{\lambda_j\left(e^{-t}-1\right)} && \text{(Fact 3.3.13)} \\
&= e^{\sum_{j=1}^{n}\lambda_j\left(e^{-t}-1\right)} = e^{\mu\left(e^{-t}-1\right)}.
\end{aligned}$$

Combining this with (3.6) and substituting $t = -\ln(1-w)$ we have obtain

$$\begin{aligned}
P\left[Y \leq (1-w)\mu\right] &\leq \frac{e^{\mu\left(e^{-t}-1\right)}}{\exp\left(-t(1-w)\mu\right)} \\
&= \frac{e^{\mu\left(e^{\ln(1-w)}-1\right)}}{\exp\left((1-w)\mu\ln(1-w)\right)} \\
&= \left(\frac{e^{-w}}{(1-w)^{(1-w)}}\right)^{\mu}
\end{aligned}$$

which is our desired result. □

We can now use Theorem 3.3.14 to help determine how foolish it would be for me to risk my house on a bet that I can get more than 40 tails in 100 fair coin flips.[14]

[14] Of course I shouldn't risk anything unless I actually get something good if I win the bet!!! Let's suppose I am offered a free cup of coffee if I win the bet. That said, sometimes people risk things on dares even if they get nothing tangible by winning the dare beyond, perhaps, perceived reputational benefits. Worse still, the bigger the risk, the more likely they are to sometimes take on the dare. In short: people are weird – thankfully this is not a psychology textbook!

Example 3.3.15 (Should I Bet My House?)**.** *Given $n = 100$ fair coin flips $I_1, \ldots, I_{100} \in \{0, 1\}$ i.i.d. random variables with densities $p(x) = \frac{1}{2}\delta(x-1) + \frac{1}{2}\delta(x)$, we want to know how likely it is that I will get more than 40 "1's" to win the bet. Note that this will happen exactly if $Y = \sum_{j=1}^{100} I_j > 40$. Given that $\mu = \mathbb{E}[Y] = 50$, we can see that I will loose the bet only if $Y \leq 40 = 0.8\mu$. Thus, we may apply Theorem 3.3.14 with $1 - w = 0.8$ in order to learn that*

$$\mathbb{P}[\text{I Loose the Bet}] = \mathbb{P}[Y \leq 0.8\mu] \leq \left(\frac{e^{-0.2}}{0.8^{0.8}}\right)^{50} < 0.342.$$

So, I will probably win the bet (with probability at least 0.658), but I can't guarantee I'm not giving my house away about every third time I try based on this analysis. I don't think I'll take the bet.

Exercise 3.3.15. *Suppose I offer to buy you a sandwich if you can flip more than 10 "Heads" in 100 fair coin flips. If you flip fewer than 10 heads, I get to take your house (or, your most valuable possession, whatever that may be). How likely are you to win? Will you take my bet?*

In fact with some additional effort one can also obtain a 2-sided version of the last theorem. It tells us in the context of our coin flipping example that we shouldn't ever bet on being able to get many fewer or many more "Tails" or "Heads" than what we expect in a large number of tries.

Theorem 3.3.16 (A Two-Sided Chernoff Inequality [52, Exercise 2.3.5])**.** *Let $X_1, \cdots, X_n \in \mathbb{R}$ be independent random variables with probability densities $p_j(x) = \lambda_j\delta(x-1)+(1-\lambda_j)\delta(x)$ where $\lambda_j \in (0, 1)$. Let $w \in (0, 1]$ and $Y = \sum_{j=1}^{n} X_j$ with $\mu := \mathbb{E}[Y] = \sum_{j=1}^{n} \lambda_j$. Then,*

$$\mathbb{P}[|Y - \mu| \geq w\mu] \leq 2e^{-c\mu w^2},$$

where $c > 0$ is an absolute constant.

Of course, we are far less interested in flipping coins herein than in computing. The next section will now give you a great example of how what we have learned so far can help us do exactly that.

3.4 Application: Fault-Tolerant Monte Carlo Integration

MARK IS WORKING HERE ABOUTS... BELOW IS ROUGHER, BUT KEEP READING! YOU WILL SURVIVE!

MEDIAN OF MEANS = ROBUSTNESS TO adversarial outliers

– Function of many variables that we can evaluate anywhere we want to the domain. No closed form or simple formula. But, each function evaluation takes a relatively long time

(e.g., a second). – Median of Means estimator will allow us to control the probability that our error is larger than a small fraction of the functions total energy... This same estimation strategy will then be utilized several more times in different applications later one.

– Define $\|f\|_2$ norm!!!!

We now turn to an algorithmic application involving these probabilistic ideas and results: Monte Carlo integration. This is useful in cases where f has no closed form expression.

Suppose $f : [0,1]^n \to \mathbb{R}$ and f can be evaluated at points.

Goal. *Evaluate*

$$\int_{[0,1]^n} f(x)dx =: Int(f)$$

Note. *Standard techniques for griding* $[0,1]^n$ *require* c^n *points for* $n \geq 100$

One very famous approach to solving this problem is a probabilistic technique called the *Monte-Carlo Integration or Method*. The power of this technique lies in the the fact that it doesn't require knowledge of the function f on it domain. A few random points suffice.

Let $f : [0,1]^N \to \mathbb{R}$. Choose $X_1, \ldots, X_m \in [0,1]^N$ are i.i.d uniform random variables. We seek to estimate the integral,

$$J := \int_{[0,1]^n} f(x)dx \tag{3.7}$$

To begin estimating this, we introduce the random variable Z, which is the sum of function evaluations at the i.i.d. uniform points X_j

$$Z := \frac{1}{m}\sum_{j=1}^{m} f(X_j). \tag{3.8}$$

By Theorem 3.3.5 we can see that

$$\mathbb{E}[Z] = \mathbb{E}[f(X_j)] = \int_{[0,1]^n} f(y)dy = J$$

Furthermore, one can show that

$$\mathrm{Var}[Z] \leq \frac{1}{m}\|f\|_2^2. \tag{3.9}$$

Exercise 3.4.1. *Prove that* (3.9) *holds.*

Lemma 3.4.1. *Choose* $\epsilon > 0$. *If* $m \geq \frac{10}{\epsilon^2}$ *then*

$$\mathbb{P}[|Z - J| \leq \epsilon\|f\|_2] \geq 0.9$$

where J *is defined as in* (3.7) *and* Z *is defined as in* (3.8).

Proof. Note from complementary events we have

$$\mathbb{P}\left[|Z-J|<\epsilon\|f\|_2\right]=1-\mathbb{P}\left[|Z-J|\geq\epsilon\|f\|_2\right]$$

As we saw in the previous discussion, we have that $|Z-J|=|Z-\mathbb{E}[Z]|$ and $\|f\|_2\leq(m\text{Var}[Z])^{1/2}$. Thus after noting $\mathbb{R}\setminus[J-\epsilon\|f\|_2,J+\epsilon\|f\|_2]\subset\mathbb{R}\setminus[J-\epsilon\,(m\text{Var}[Z])^{1/2},J+(m\text{Var}[Z])^{1/2}]$ apply Chebyshev's inequality and the hypothesis to obtain

$$\mathbb{P}\left[|Z-J|\geq\epsilon\|f\|_2\right]\leq\mathbb{P}\left[|Z-J|\geq\epsilon\,(m\text{Var}[Z])^{1/2}\right]\leq\frac{1}{m\epsilon^2}\leq\frac{1}{\left(\frac{10}{\epsilon^2}\right)\epsilon^2}=0.1$$

So in turn using the complement of the event $|Z-J|\geq\epsilon\|f\|_2$, we have the desired result. □

Using Lemma 3.4.1, we can justify the statement that error of Monte Carlo Integration decays $\mathcal{O}(\frac{1}{\sqrt{m}})$. That is, solving for ϵ in the hypothesis of Lemma 3.4.1 we have $\epsilon=\sqrt{\frac{10}{m}}$.

We now turn to a method by which we can increase the likelihood of a our estimate being within a desired error bound beyond the guarantee seen in Lemma 3.4.1. To that end we introduce some notation, building several independent Z estimators as in (3.8).

Let

$$Z_k:=\frac{1}{m}\sum_{j=1}^{m}f(X_{k,j})\tag{3.10}$$

where $X_{k,j}\in[0,1]^N, 1\leq j\leq m, 1\leq k\leq K$ are all i.i.d uniform random variables (say we are double indexing mK random varaibles...). We have described repeating the experiment K times and gathering these estimators in the independent random variables $\{Z_k\}_{k=1}^K$. We then use the median of these estimators:

Furthermore, let

$$I_k:=\begin{cases}1 & |Z_k-J|\leq\epsilon\|f\|_2\\ 0 & \text{otherwise}\end{cases}\tag{3.11}$$

These are indicator functions for whether a given estimate Z_k is within the desired error bound ϵ to the target integral J. Note that since the random variables Z_k are independent, then so are the discrete random variables I_k.

How big then does K need to be to achieve some desired likelihood q of a sufficiently accurate estimator $\tilde{Z}$ (estimator is defined in equation 3.12)? The following Lemma provides an answer: when K is larger than $C\log\frac{1}{q}$, most of our estimates will be accurate within the error bound with probability at least $1-q$.

Lemma 3.4.2. *Let I_k for $k\in[K]$ be independent indicator variables as defined in* (3.11). *$\exists C\in[0,\infty)$ so that when $K\geq C\log\frac{1}{q}$ then*

$$\mathbb{P}\left[\sum_{k=1}^{K}I_k\leq K/2\right]\leq q$$

for any arbitrary $q \in (0,1)$, provided $m \geq 10/\epsilon^2$. Here C is a constant that is independent of all other quantities (and in fact is bounded by log.95 thing)...

Proof. Lemma 3.4.1 implies that $\tilde{p} = \mathbb{P}[I_k = 1] \geq 0.9$. Theorem 3.3.14 however provides us a means to bound the sum of indicator variables of this sort. Noting $K/2 = (1-(1-1/2\tilde{p}))K\tilde{p}$ and $\mathbb{E}\left[\sum_{k=1}^{K} I_k\right] = K\tilde{p}$ we have that

$$\begin{aligned}
\mathbb{P}\left[\sum_{k=1}^{K} I_k \leq K/2\right] &= \mathbb{P}\left[\sum_{k=1}^{K} I_k \leq (1-(1-1/2\tilde{p}))K\tilde{p}\right] \\
&\leq \left[\frac{\exp\left(-(1-2/\tilde{p})\right)}{(1/2\tilde{p})^{(1/2\tilde{p})}}\right]^{K\tilde{p}} \\
&= \left(\sqrt{2\tilde{p}}\exp\left(-(\tilde{p}-1/2)\right)\right)^{K} \\
&\leq \left(\sqrt{2}e^{-0.4}\right)^{K} \\
&\leq 0.95^{K}
\end{aligned}$$

where we have used $w \leftarrow (1-1/2\tilde{p})$ in Theorem 3.3.14. So

$$0.95^{K} \leq q \implies K \geq -\log(0.95)\log(1/q)$$

and we have the desired result. □

Theorem 3.4.3 (Median of Means Estimation for Monte Carlo)**.** *Let $\epsilon, q \in (0,1)$ and define Z_k as in 3.10 with $m \geq 10/\epsilon^2$ and K an odd integer such that $K \geq C\log(1/q)$ where C is the constant from Lemma 3.4.2. Let $J = \int_{[0,1]^N} f(x)dx$ for $f : [0,1]^N \to \mathbb{R}$. Define*

$$\tilde{Z} := \textit{median}\{Z_1, \ldots, Z_K\}. \tag{3.12}$$

Then,

$$\left|\tilde{Z} - J\right| \leq \epsilon\|f\|_2 \tag{3.13}$$

hold with probability at least $(1-q)$. The total required number of function evaluations of f is $\mathcal{O}\left(\log(1/q)/\epsilon^2\right)$

Proof. The proof follows from Lemmas 3.4.1 and 3.4.2 along with Lemma 3.4.4, and is left as an exercise. □

Lemma 3.4.4. *Let $K \in \mathbb{N}$ be odd, and I_k for $k \in [K]$ be independent indicator variables as defined in* (3.11)*. If $\sum_{k=1}^{K} I_k > \frac{K}{2}$, then $\tilde{Z}$ in* (3.12) *satisfies* (3.13)*.*

Proof. Assume $\sum_{k=1}^{K} I_k > \frac{K}{2}$, and for the purposes of contradiction, further suppose that $\tilde{Z} < \text{Int}(f) - \epsilon\|f\|_2$. Because K is odd, $\tilde{Z} \in \{Z_1, \ldots, Z_K\}$. Hence, $\tilde{Z} < \text{Int}(f) - \epsilon\|f\|_2$ implies that at least $\frac{K-1}{2} + 1$ elements of the set $\{Z_1, \ldots, Z_K\}$ are less than $\text{Int}(f) - \epsilon\|f\|_2$, and thus their corresponding indicator variables I_k are all zero. However $\frac{K-1}{2} + 1 > \frac{K}{2}$, so in this case $\sum_{k=1}^{K} I_k < \frac{K}{2}$ must hold, contradicting our initial assumption. Thus, $\tilde{Z} \leq \text{Int}(f) - \epsilon\|f\|_2$ cannot hold. Similarly, we will arrive at another contradiction should we assume $\tilde{Z} > \text{Int}(f) + \epsilon\|f\|_2$. □

Exercise 3.4.2. *Re-prove Lemma 3.4.1 for $f : [0,1]^n \to \mathbb{C}$.*
HINT: If $A, B \in \mathbb{R}$ are random numbers, then $\mu = \mathbb{E}[A + \mathbb{i}B] = \mathbb{E}[A] + \mathbb{i}\mathbb{E}[B]$ and $\text{Var}[A + \mathbb{i}B] = \mathbb{E}[(A + \mathbb{i}B - \mu)\overline{(A + \mathbb{i}B - \mu)}]$.

Exercise 3.4.3. *Write out a formal proof of Theorem 3.4.3.*

3.5 Conditional Probability

Why is conditional probability useful for computation? It's sometimes easier to compute/bound a probability one is interested using conditional expectations than it is to directly compute the probability. We will see examples of this later in the LSH section.

Definition 3.5.1. *The conditional probability that $X \in \mathcal{T} \subseteq \mathbb{R}^n$ given that $Y \in \mathcal{S} \subseteq \mathbb{R}^m$ is defined to be*

$$\mathbb{P}\left[X \in \mathcal{T} \;\middle|\; Y \in \mathcal{S}\right] := \frac{\mathbb{P}[X \in \mathcal{T} \;\&\; Y \in \mathcal{S}]}{\mathbb{P}[Y \in \mathcal{S}]} = \frac{\mathbb{P}[(X,Y) \in \mathcal{T} \times \mathcal{S}]}{\mathbb{P}[Y \in \mathcal{S}]}$$

provided that $\mathbb{P}[Y \in \mathcal{S}] > 0$.

Example 3.5.2. *Let $X \sim \mathcal{N}(0, \sigma^2)$ and $b > a \geq 0$. The density function of X is $p_X(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$. Then*

$$\begin{aligned}
\mathbb{P}\left[X < 0 \;\middle|\; |X| \in (a,b)\right] &= \frac{\mathbb{P}\left[X \in (-b,-a)\right]}{\mathbb{P}\left[X \in (a,b) \bigcup (-b,-a)\right]} \\
&= \frac{\frac{1}{\sqrt{2\pi}\sigma} \int_{-b}^{-a} e^{-\frac{y^2}{2\sigma^2}} dy}{\frac{1}{\sqrt{2\pi}\sigma}\left(\int_{-b}^{-a} e^{-\frac{y^2}{2\sigma^2}} dy + \int_{a}^{b} e^{-\frac{y^2}{2\sigma^2}} dy\right)} \\
&= \frac{1}{2}
\end{aligned}$$

since $e^{-\frac{y^2}{2\sigma^2}}$ is an even function.

Note that this computation conforms to our intuition: if the probability density of a random number is symmetric about the y-axis, then knowing something about its absolute value shouldn't tell us anything about its sign. In the next example, however, the density is not symmetric so that we can infer one sign is indeed more likely when given absolute value information.

Example 3.5.3. *Let $X \sim \text{Unif}([-2,4])$ and denote the indicator function on a given set $\mathcal{S} \subset \mathbb{R}$ by*

$$\chi_{\mathcal{S}}(y) := \begin{cases} 1 & \text{if } y \in \mathcal{S} \\ 0 & \text{if } y \notin \mathcal{S} \end{cases}.$$

Then,

$$\begin{aligned} \mathbb{P}\left[X < 0 \;\middle|\; |X| \in (1,4)\right] &= \frac{\mathbb{P}\left[X \in (-4,-1)\right]}{\mathbb{P}\left[X \in (1,4) \bigcup (-4,-1)\right]} \\ &= \frac{\frac{1}{6}\int_{-4}^{-1} \chi_{[-2,4]}(y)\, dy}{\frac{1}{6}\int_{1}^{4} \chi_{[-2,4]}(y)\, dy + \frac{1}{6}\int_{-4}^{-1} \chi_{[-2,4]}(y)\, dy} \\ &= \frac{1}{4}. \end{aligned}$$

Exercise 3.5.1. *Let $X \sim \text{Unif}([-2,4])$. Compute the following conditional probabilities.*

(a) Show that $\mathbb{P}\left[X < 0 \;\middle|\; X^2 \in (1,4)\right] = \frac{1}{2}$.

(b) Show that $\mathbb{P}\left[X^2 \in (1,4) \;\middle|\; X < 0\right] = \frac{1}{2}$.

(c) Show that $\mathbb{P}\left[X^2 \in (1,4) \;\middle|\; X > 0\right] = \frac{1}{4}$.

(d) Compute $\mathbb{P}\left[X^2 \in (0,4) \;\middle|\; |X| < 1\right]$.

(e) Compute $\mathbb{P}\left[X < 0 \;\middle|\; X^2 > 4\right]$.

(f) Compute $\mathbb{P}\left[X > 0 \;\middle|\; X^2 < 4\right]$.

Exercise 3.5.2. *Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ be independent random vectors. Show that $\mathbb{P}\left[X \in \mathcal{T} \;\middle|\; Y \in \mathcal{S}\right] = \mathbb{P}[X \in \mathcal{T}]$ holds for all reasonable $\mathcal{S} \subset \mathbb{R}^m$ and $\mathcal{T} \subset \mathbb{R}^n$.*

Theorem 3.5.4 (Baye's Law)**.** *If $\mathbb{P}\left[Y \in \mathcal{S}\right] \neq 0$ and $\mathbb{P}\left[X \in \mathcal{T}\right] \neq 0$ both hold, then*

$$\mathbb{P}\left[X \in \mathcal{T} \;\middle|\; Y \in \mathcal{S}\right] = \frac{\mathbb{P}\left[Y \in \mathcal{S} \;\middle|\; X \in \mathcal{T}\right] \cdot \mathbb{P}\left[X \in \mathcal{T}\right]}{\mathbb{P}\left[Y \in \mathcal{S}\right]}.$$

Exercise 3.5.3. *Prove Baye's Law.*

As we will see later in, e.g., Section 3.7, conditional probability can be a valuable tool in helping us to bound the probability that complicated functions of a random variable behave as we expect. Before we can see this useful tool in action, however, we will need to justify some commonly used notation. Toward this end, let $\mathcal{S} = (a,b) \times (c,d) \subset \mathbb{R}^2$ be the cartesian product of two open intervals, choose $z \in (a,b)$, and consider two independent random numbers $X, Y \in \mathbb{R}$ with densities p_X and p_Y, respectively. In this case we will define make it clear that we have a division by zero if we consider the "event" X = z, so we use limits to say what the notation means... this is 0/0 unless we make sense of it more carefully.

$$\begin{aligned}
\mathbb{P}\left[(X,Y) \in \mathcal{S} \;\middle|\; X = z\right] &:= \lim_{\epsilon \to 0} \mathbb{P}\left[(X,Y) \in \mathcal{S} \;\middle|\; X \in (z-\epsilon, z+\epsilon)\right] \\
&= \lim_{\epsilon \to 0} \frac{\mathbb{P}\left[(X,Y) \in \mathcal{S} \;\&\; X \in (z-\epsilon, z+\epsilon)\right]}{\mathbb{P}\left[X \in (z-\epsilon, z+\epsilon)\right]} \\
&= \lim_{\epsilon \to 0} \frac{\mathbb{P}\left[(X,Y) \in (z-\epsilon, z+\epsilon) \times (c,d)\right]}{\mathbb{P}\left[X \in (z-\epsilon, z+\epsilon)\right]} \\
&= \lim_{\epsilon \to 0} \frac{\int_{z-\epsilon}^{z+\epsilon} \int_c^d p_X(u) p_Y(v) \; dv du}{\int_{z-\epsilon}^{z+\epsilon} p_X(u) \; du} \\
&= \int_c^d p_Y(v) \; dv.
\end{aligned}$$

Using this calculation motivation we can now define the following more general notation.

Definition 3.5.5. *Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ be independent random vectors, $\mathcal{S} \subseteq \mathbb{R}^{n+m}$, and $\mathbf{z} \in \mathbb{R}^n$. Then,*

$$\mathbb{P}\left[(X,Y) \in \mathcal{S} \;\middle|\; X = \mathbf{z}\right] := \int_{\mathcal{S}_{\mathbf{z}} := \{\mathbf{y} | (\mathbf{z},\mathbf{y}) \in \mathcal{S}\}} p_Y(\mathbf{v}) \; d\mathbf{v},$$

where p_Y is the density of Y.

With Definition 3.5.5 in hand we can now prove a useful lemma which will have several applications below.

Lemma 3.5.6. *Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ be independent random vectors with densities p_X and p_Y, respectively. Then if $f : \mathbb{R}^{n+m} \to \mathbb{R}^d$, $\mathcal{S} \subseteq \mathbb{R}^d$, and $\mathcal{T} \subseteq \mathbb{R}^n$, we have that*

$$P\left[f(X,Y) \in \mathcal{S}\right] = \mathbb{P}\left[f(X,Y) \in \mathcal{S} \;\&\; X \in \mathcal{T}\right] = \int_{\mathcal{T}} \mathbb{P}\left[f(X,Y) \in \mathcal{S} \;\middle|\; X = \mathbf{z}\right] p_X(\mathbf{z}) \; d\mathbf{z}$$

whenever $f^{-1}(\mathcal{S}) \subseteq \mathcal{T} \times \mathbb{R}^m$.

Proof. Beginning with the left most expression we have that

$$\begin{aligned}\mathbb{P}\left[f(X,Y)\in\mathcal{S}\right] &= \mathbb{P}\left[(X,Y)\in f^{-1}(\mathcal{S})\right]\\ &= \mathbb{P}\left[(X,Y)\in f^{-1}(\mathcal{S})\cap(\mathcal{T}\times\mathbb{R}^m)\right]\\ &= \mathbb{P}\left[f(X,Y)\in\mathcal{S}\ \&\ X\in\mathcal{T}\right]\end{aligned}$$

where we have used that $f^{-1}(\mathcal{S}) = f^{-1}(\mathcal{S})\cap(\mathcal{T}\times\mathbb{R}^m)$ in the last line. Continuing our formal calculation we now further see that

$$\begin{aligned}\mathbb{P}\left[f(X,Y)\in\mathcal{S}\ \&\ X\in\mathcal{T}\right] &= \mathbb{P}\left[(X,Y)\in f^{-1}(\mathcal{S})\cap(\mathcal{T}\times\mathbb{R}^m)\right]\\ &= \int_{\mathcal{T}}\left(\int_{\{\mathbf{y}|(\mathbf{u},\mathbf{y})\in f^{-1}(\mathcal{S})\}} p_Y(\mathbf{v})\ d\mathbf{v}\right) p_X(\mathbf{u})\ d\mathbf{u},\end{aligned}$$

where we have again used that $f^{-1}(\mathcal{S})\subseteq\mathcal{T}\times\mathbb{R}^m$ on the last line. Substituting Definition 3.5.5 into the last expression now yields the desired result. □

We are now in the position to use Lemma 3.5.6 to prove one of the most fundamental properties of Gaussian random variables.

3.5.1 Linear Combinations of Independent Gaussians are Gaussian

Perhaps one of the most fundamental properties of Gaussian random variables is that they are closed under linear transformations (i.e., linear combinations of independent Gaussian random variables are again Gaussian random variables). This is a special property that does not hold for random variables in general. Consider, e.g., two i.i.d. coin flips $X_1, X_2 \in \{0,1\}$. The sum of two such random variables is certainly no longer a random coin flip given that, e.g., $X_1 + X_2$ now has a nonzero probability of being 2. More generally, one can convince themselves using similar arguments that no discrete random variable will ever be closed under arbitrary linear combinations of i.i.d. copies. Indeed, the Central Limit Theorem effectively guarantees that large finite linear combinations of i.i.d. copies of general classes of continuous random variables will no longer have the same distribution either (because, e.g., they change to become "more Guassian" when averaged). Independent gaussian random variables, on the other hand, do yield new Gaussian random variables under linear combinations. The next lemma proves that this remarkable property holds.

Lemma 3.5.7. *Let $X\sim N(0,\sigma_x)$ and $Y\sim N(0,\sigma_y)$ be two independent mean 0 Gaussian random numbers with variance σ_x^2 and σ_y^2 respectively. Then*

$$X+Y\sim N\left(0,\sqrt{\sigma_x^2+\sigma_y^2}\right).$$

Proof. It suffices to show for an arbitrary interval $\mathcal{S} = (a, b)$ that

$$P\left[X + Y \in \mathcal{S}\right] = \frac{1}{\sqrt{2\pi(\sigma_x^2 + \sigma_y^2)}} \int_a^b \exp\left(-\frac{x^2}{2(\sigma_x^2 + \sigma_y^2)}\right) \, dx. \tag{3.14}$$

To start, we consider $\mathcal{T} = \mathbb{R}$ and apply Lemma 3.5.6 with $f(x, y) = x + y$ to obtain

$$P\left[X + Y \in \mathcal{S}\right] = P\left[X + Y \in \mathcal{S} \ \& \ X \in \mathbb{R}\right] = \int_{\mathbb{R}} \mathbb{P}\left[X + Y \in \mathcal{S} \mid X = x\right] p_X(x) \, dx.$$

Utilizing Definition 3.5.5 in our last expression after noting that $\mathcal{S}_x = \{y \in \mathbb{R} \mid f(x, y) \in (a, b)\} = (a - x, b - x)$ we have that

$$\begin{aligned} P\left[X + Y \in \mathcal{S}\right] &= \int_{\mathbb{R}} \left(\int_{\mathcal{S}_x} p_Y(y) \, dy \right) p_X(x) \, dx \\ &= \frac{1}{2\pi\sigma_x\sigma_y} \int_{\mathbb{R}} \int_{a-x}^{b-x} \exp\left(-\frac{y^2}{2\sigma_y^2}\right) \exp\left(-\frac{x^2}{2\sigma_x^2}\right) dy dx \\ &= \frac{1}{2\pi\sigma_x\sigma_y} \int_{\mathbb{R}} \int_a^b \exp\left(-\frac{(z-x)^2}{2\sigma_y^2}\right) \exp\left(-\frac{x^2}{2\sigma_x^2}\right) dz dx. \end{aligned}$$

In the last line above we used the change of variable $z = x + y$ in the inner integral. In fact, if you continue by combining and then simplifying the exponentials in the integral above, and then utilize another clever substitution, you will now be able to finishing proving that (3.14) holds yourself. See Exercise 3.5.4. □

Exercise 3.5.4. *Finish the proof of Lemma 3.5.7.*

We can now use Lemma 3.5.7 to better understand the inner product of a Gaussian random vector with another fixed vector $\mathbf{x}$. In the following exercises you will prove that if $\mathbf{g} \sim N(\mathbf{0}, I_n)$, then $\langle \mathbf{g}, \mathbf{x} \rangle \sim N(0, \|\mathbf{x}\|_2)$. This fact will be important in Chapter 4.

Exercise 3.5.5. *Let $X \sim N(0, \sigma)$. Show that $aX \sim N(0, |a|\sigma)$ $\forall a \in \mathbb{R} \setminus \{0\}$ by*

(a) first proving that that $aX \sim N(0, a\sigma)$ $\forall a \in (0, \infty)$, and

(b) then by proving that $-aX \sim N(0, a\sigma)$ $\forall a \in (0, \infty)$.

Exercise 3.5.6. *Let $\mathbf{g} \sim N(\mathbf{0}, I_n)$ and $\mathbf{x} \in \mathbb{R}^n$. Use Lemma 3.5.7 along with the last exercise to prove that $\langle \mathbf{g}, \mathbf{x} \rangle \sim N(0, \|\mathbf{x}\|_2)$.*

3.6 Application: Markov Chains and Morris's Algorithm

3.6.1 Problem Statement and the Naive Solution

Talk about streaming model. Can't record every conversation you have with every person over your entire life – we take. notes to summarize. You can't store all the internet traffic that ever happens. Want to summarrize constant stream of internet data.

Counting objects is a common challenge in settings involving very large data sets. Memory efficient methods are needed in order to make object counts feasible for routine use on these data. This type of problem and the ensuing discussion will also serve as an introduction to some key ideas for the course. In it we see a deterministic, simple sounding task (counting in this case) which under further study shows the need for fast and memory efficient algorithms that give good approximations to well constructed statistics questions.

A formal statement of the problem is as follows: Given a sequence $\{z_j\}_{j=1}^N$ where $\forall j$, $z_j \in U$ and some item $w \in U$ of interest, count the number of occurrences of term w in the sequence $\{z_j\}_{j=1}^N$.

Goal. *Estimate the count of w occurring in the sequence using $\lceil \log_2 \lceil \log_2 N \rceil \rceil$ bits of memory. We require our estimate of the count be larger than the actual count, but no more than twice the actual count.*

The source of the overestimate error on the count will be made clear shortly. Examples abound for data sets for which counts of this sort are useful

Example 3.6.1. *U is the set of all possible phone numbers, and $\{z_j\}_{j=1}^N$ is a list of phone numbers which have communicated with a particular cellphone tower over some period of time. The term w is a phone number of interest, perhaps a known spammer.*

Example 3.6.2. *U is all possible pairs of words in the English language. So* `hello world` *or* `thank you` *are members of U. The sequence $\{z_j\}_{j=1}^N$ is a list of all pairs of words that appear in emails contained in some user's inbox. The term w then could be "buy Ford" which is of interest to perhaps stock traders or advertisers.*

Example 3.6.3. *U is all possible IP addresses and $\{z_j\}_{j=1}^N$ is a list that contains the originating IP address for all packets received by a certain router. The term w is the IP address of a server used by a movie streaming service of interest to an internet service provider.*

We may wish to consider counts of many different terms w for say all cell-phone towers in a particular country, or all users of some particular email service. Clearly, the size of such data sets means that counts can be potentially very large. Since N is an integer, a priori, we would need (maximally) $\lfloor \log_2 N \rfloor + 1$ bits to store a count of each w.

Note. *We can store N using $\lfloor \log_2 N \rfloor + 1$ bits. We have $\lfloor \log_2 N \rfloor = k$ only if $k \leq \log_2 N < k+1$ if and only if $2^k \leq N < 2^{k+1}$. That is, $2^k \leq N < 2^{k+1}$ is the range of integers which requires $k+1$ bits. So the integers requiring 4-bits for example are 8 through 15. Depending on implementation there are other bits required to say, store the sign of the integer. For simplicity we say that storing an integer of size N requires $\lceil \log_2 N \rceil$ bits, though this may be off by one, or some other constant, depending on implementation.*

A first, naive approach is to increment a counter after one scan of the sequence, and then store the logarithm of that count.

Algorithm 15 Naive Counter

Input: $\{z_j\}_{j=1}^N, w$
Output: approximate count of w in $\{z_j\}_{j=1}^N$
for $j = 1$ to N **do**
  if $z_j = w$ **then**
    $\tilde{w} \leftarrow \tilde{w} + 1$
  end if
end for
$E \leftarrow \lceil \log_2 \tilde{w} \rceil$

Since E is of size at most $\lceil \log_2 N \rceil$ it takes at most $\lceil \log_2 \lceil \log_2 N \rceil \rceil$ bits to store. Due to the information lost by taking the ceiling, we also have that $\tilde{w} \leq 2^E \leq 2\tilde{w}$.

Does E and the algorithm 15 achieve our goal?

No. While it is true that E occupies the right number of bits, the counter itself $\tilde{w}$ would need to occupy possibly $\lceil \log_2 N \rceil$ bits when running the algorithm.

3.6.2 A Lower-Memory Solution

Given a sequence of $z_0, \ldots, z_{n-1} \in \{0, 1\}$ count the number of times that a 1 appears in the sequence. We would like to use only $c_1 \lceil \log \lceil \log n \rceil \rceil$ number of bits to store our estimate of the counting problem, where c_1 is an absolute constant (independent of n).

We introduce some notation:

- Let $T_j = \sum_{\ell=1}^{j} z_\ell$. This is the true count of the one's in the given sequence.
- $\forall j \in [n]$ output X_j such that $|X_j - T_j| \leq c_2 T_j$ where c_2 should be a (small) absolute constant independent of j.

In algorithm form we have:

Algorithm 16 Morris' Algorithm

Input: $z_0, z_1, \ldots, z_{n-1} \in \{0,1\}$
Output: $X_j \approx T_j = \sum_{\ell=0}^{j} z_j$, $\forall j \in [n]$

$Y_{-1} = 0$
for $j = 1, \ldots, n-1$ **do**
  if $z_j = 0$ **then**
    $Y_j \leftarrow Y_{j-1}$
  else
    $B \sim \begin{cases} 1 & \text{with probability } 2^{-Y_{j-1}} \\ 0 & \text{with probability } 1 - 2^{-Y_{j-1}} \end{cases}$
    $Y_j \leftarrow Y_{j-1} + B$
  end if
  $X_j \leftarrow 2^{Y_j} - 1$
end for

To see why the memory usage fits our stated goal, consider $2^{Y_{n-1}} - 1 \leq c_2 T_{n-1}$ which implies $Y_{n-1} \leq \log_2(c_2 T_{n-1} + 1)$ which takes $\lceil \log \lceil \log(c_2 T_{n-1} + 1) \rceil \rceil$.

Note that $Y_0, \ldots, Y_{n-1}$ is an example of Markov chain because the process is "memoryless" – the next state only depends on the current state.

In order to simplify analysis, we will consider the subsequences that correspond to the actual events of interest: Let $z_{i_1}, \ldots, z_{i_{\tilde{n}}}$ be the members of the sequence where $z_j = 1$. We denote then $\tilde{Y}_k = Y_{j_k}$ for $k = 1, \ldots, \tilde{n} - 1$. This corresponds then to our estimates at the different points in the stream where the event of interest has occured.

In practice generating the random variable B with accuracy that accounts for potentially very small values $2^{-Y_{j-1}}$ itself could torpedo the project of making a low bit counter – however the efficient generation of random numbers with high accuracy is a involved topic outside our scope. In this course we'll take it for granted that it can be accomplished.

For the following Lemmas we take the random variables to be defined as described in Algorithm 16

3.6.3 Analysis of Morris's algorithm

To show that $\left| 2^{Y_{j-1}} - 1 - T_j \right| \leq c T_j$ holds it's good enough to analyze the subsequence $\tilde{Y}_1 = Y_{i_1}, \tilde{Y}_2 = Y_{i_2}, \cdots$ for $i'_m s$ where $z_{i_m} = 1$. We denote $\tilde{X}_j = 2^{\tilde{Y}_{j-1}} - 1$.

Lemma 3.6.4. *Let $m, j \in \mathbb{N}$ be such that $m \geq 1$ and $j \geq 0$. Then:*

$$\mathbb{E}\left[2^{m\tilde{Y}_j}\right] = (2^m - 1)\mathbb{E}\left[2^{(m-1)\tilde{Y}_{j-1}}\right] + \mathbb{E}\left[2^{m\tilde{Y}_{j-1}}\right].$$

Proof.

$$\begin{aligned}
\mathbb{E}[2^{m\tilde{Y}_j}] &= \sum_{i\in\mathbb{Z}} 2^{mi}\mathbb{P}[\tilde{Y}_j = i] \\
&= \sum_{i\in\mathbb{Z}} 2^{mi}\left[\frac{1}{2^{i-1}}\mathbb{P}[\tilde{Y}_{j-1} = i-1] + \left(1-\frac{1}{2^i}\right)\mathbb{P}[\tilde{Y}_{j-1} = i]\right] \\
&= \sum_{i\in\mathbb{Z}}\left\{2\cdot 2^{(m-1)i}\mathbb{P}\left[\tilde{Y}_{j-1} = i-1\right] + 2^{mi}\mathbb{P}\left[\tilde{Y}_{j-1} = i\right] - 2^{(m-1)i}\mathbb{P}\left[\tilde{Y}_{j-1} = i\right]\right\} \\
&= \sum_{i\in\mathbb{Z}} 2^m\cdot 2^{(m-1)(i-1)}\mathbb{P}\left[\tilde{Y}_{j-1} = i-1\right] + \sum_{i\in\mathbb{Z}}\left\{2^{mi} - 2^{(m-1)i}\right\}\mathbb{P}\left[\tilde{Y}_{j-1} = i\right] \\
&= 2^m\mathbb{E}[2^{(m-1)\tilde{Y}_{j-1}}] + \mathbb{E}\left[2^{m\tilde{Y}_{j-1}} - 2^{(m-1)\tilde{Y}_{j-1}}\right] \\
&= (2^m-1)\mathbb{E}[2^{(m-1)\tilde{Y}_{j-1}}] + \mathbb{E}[2^{m\tilde{Y}_{j-1}}].
\end{aligned}$$

□

Lemma 3.6.5 (CMSE 890 Lecture 4)**.**

$$\mathbb{E}\left[\tilde{X}_j\right] = j \qquad \forall j = 0, 1, \cdots \tilde{n} \leq n.$$

Proof.

$$\mathbb{E}[\tilde{X}_j] = \mathbb{E}[2^{\tilde{Y}_1} - 1] = \mathbb{E}[2^{\tilde{Y}_j}] - 1.$$

So it suffices to show that $\mathbb{E}[2^{\tilde{Y}_j}] = j + 1$.
We can show this using induction.

- Base Case: $\mathbb{E}[2^{\tilde{Y}_0}] = 1 = 0 + 1$.
- inductive hypothesis: Suppose $\mathbb{E}[2^{\tilde{Y}_{j-1}}] = j$.
- Then setting $m = 1$ in Lemma 3.6.4 we get:

$$\mathbb{E}[2^{\tilde{Y}_j}] = \mathbb{E}[1] + \mathbb{E}[2^{\tilde{Y}_{j-1}}] = j + 1.$$

□

Lemma 3.6.6. $\mathbb{E}[2^{2\tilde{Y}_j}] = \frac{3}{2}j^2 + \frac{3}{2}j + 1$.

Proof.

$$\begin{aligned}
&\mathbb{E}[2^{2\tilde{Y}_j}] = 3\mathbb{E}[2^{\tilde{Y}_{j-1}}] + \mathbb{E}[2^{2j\tilde{-}1}] \quad \text{setting } m = 2 \text{ in Lemma 3.6.4} \\
&\qquad\vdots \\
&\mathbb{E}[2^{2\tilde{Y}_0}] = 3\mathbb{E}[2^{\tilde{Y}_0}] + \mathbb{E}[2^{\tilde{Y}_0}]
\end{aligned}$$

Adding all the equations and simplifying, we get:

$$\begin{aligned}\mathbb{E}[2^{2\tilde{Y}_j}] &= 3\left\{\sum_{k=0}^{j-1}\mathbb{E}[2^{\tilde{Y}_k}]\right\} + \mathbb{E}[2^{\tilde{Y}_0}] \\ &= 3\Big\{1+2+\cdots+j\Big\}+1 \\ &= \frac{3j(j+1)}{2}+1.\end{aligned}$$

□

Lemma 3.6.7. $\mathrm{Var}\left[\tilde{X}_j\right] = \frac{1}{2}(j^2-j)$.

Exercise 3.6.1. *Prove Lemma 3.6.7.*

An application of Chebychev's inequality now shows us that:

$$\begin{aligned}\mathbb{P}\left[|\tilde{X}_j - j| \geq kj\right] &= \mathbb{P}\left[|\tilde{X}_j - j| \geq \frac{kj}{\sqrt{\frac{1}{2}(j^2-j)}}\sqrt{\frac{1}{2}(j^2-j)}\right] \\ &\leq \frac{\frac{1}{2}(j^2-j)}{k^2j^2} \\ &\leq \frac{1}{2k^2}.\end{aligned}$$

Now to improve the variance we take the average L independent estimators:

$$\tilde{X}'_j := \frac{1}{L}\sum_{\ell=0}^{L-1}\tilde{X}_j^{\ell}.$$

Again Using Chebychev's inequality we have:

$$\mathbb{P}\left[|\tilde{X}'_j - j| \geq kj\right] \leq \frac{1}{2k^2L}.$$

Now in a manner similar to Lemma 3.4.1, and assuming $L \geq 5/\epsilon^2$ (we've relabeled k as ϵ) we obtain:

$$P\left[\left|\tilde{X}'_j - j\right| \geq \epsilon j\right] \leq \frac{1}{10}.$$

Also we consider the random variables:

$$I_j := \begin{cases}1 & \text{if } \left|\tilde{X}'_j - j\right| < \epsilon j \\ 0 & \text{otherwise}\end{cases}.$$

Where $\tilde{X}_j^{(i)}$ is an independent mean estimator.

As was done previously during the discussion of Monte Carlo integration, we will consider repeating the experiment of finding means and use this collection's median to estimate the desired quantity.

PROB Estimate below is broken

Indeed if $i = 0, 1, \cdots, \log(\frac{1}{q})$ for some $q \in (0,1)$ then using Chernoff's bound:

$$\mathbb{P}\left[\left|median\{\tilde{X}_j^{(i)}, \cdots, \tilde{X}_j^{(c\log(1/q))}\} < \epsilon j\right|\right] \geq (1-q)$$

3.6.4 Memory usage of Median of Means Estimators

$$\begin{aligned}\#\text{of bits used} &\leq \mathcal{O}\left(\left(\log\log n\right) \cdot L \cdot \log(\frac{1}{q})\right) \\ &= \mathcal{O}\left(\frac{1}{\epsilon^2}\log(\frac{1}{q})\log\log n\right)\end{aligned}$$

- Trivial counter was $\mathcal{O}(\log n)-$bits.
- If we expect to make $\leq \frac{t}{q}$ estimate queries for $q \in (0,1)$ then we will have the correct count up to $\epsilon-$multiplicative error with probability at least $1-t$ for all queries by the union bound.

Theorem 3.6.8 (Morris's Algorithm)**.** *Let $L = 5/\epsilon^2$ and $I = c\log(\frac{t}{q})$ where $\epsilon, q \in (0,1)$ and $c \in \mathbb{R}^+$ is an absolute constant. Then the approximate counter make clear than you modify the existimator X_j by multiplying by $1/1-\epsilon\ldots$ or something X_j resulting in a median of means approach exists such that :*

$$T_j \leq X_j \leq \left(\frac{1+\epsilon}{1-\epsilon}\right)T_j$$

holds for any t values of j with probability at least $1-q$ and the estimator X_j requires at most $\mathcal{O}(L \cdot I \cdot \log\log n)-$bits of memory with probability at least $1 - LIq$.

Exercise 3.6.2. *Use the discussion in this section to write a formal proof for Theorem 3.6.8.*

3.7 Application: Locality Sensitive Hashing and (c, r)-Nearest Neighbor Problem

3.7.1 Problem Statement and the Naive Solution

The next problem we consider is Nearest Neighbor in $\mathbb{R}$. Here we have a set of points, and are presented with a query point and wish to return the closet point in our set to the query point, reckoned by a norm of interest. Formally, we have $S \subset \mathbb{R}^D$, and query $\mathbf{y} \in \mathbb{R}^D$ and compute Define arg min $\mathbf{y}_{NN} = \arg\min \|\mathbf{x} - \mathbf{y}\|$. The set S has cardinality N which can be very large. Naturally we can extend this to k-nearest neighbors by returning a list of the k closest points.

A simple linear scan then of the set is perhaps the most obvious solution to the problem

Algorithm 17 Naive Nearest Neighbors

Input: $S, \mathbf{y}, \|\cdot\|$
Output: $\mathbf{y}_{NN}$

```
d = ∞
for x in S do
  if ||x − q|| < d then
    y_NN ← x
  end if
end for
```

This problem is a fundamental building block type of problem in many algorithms and data science applications.

Example 3.7.1. *S is the a database of gray-scale images. A query point q is a novel image, we return the image that is closest to using the ℓ_1 norm*

Example 3.7.2. *S is a database of names of people who bought departing tickets from a given airport. A query point q is a name of a passenger of interest, we return the name that is closest to it using the Hamming distance.*

Example 3.7.3. *S is a database of users of a dating website. Each user has a vector of different features, which is computed from data collected about their interests, hobbies, preferences, etc. A query point q represents a particular user, and developers for the website have engineered a norm which represents similarity between users. The closest point in S is recommended as a potential partner.*

Since each of the N points in S needs to be compared to the query point, and calculating the norm of the difference depends on the dimension D of the space, this scan has $\mathcal{O}(ND)$ complexity. We will later study how to improve on this using good approximations.

THERE are also batch NN searches where you feed in P points, and want to identify everyone's most similar neighbor. There are really $\sim P^2$ distances between all these points. We can find nearly the shortest ones despite not even looking at them all!

Recall the nearest-neighbor problem: Given $S = \{\mathbf{x_0}, \ldots, \mathbf{x_{p-1}}\} \subseteq \mathbb{R}^D$ find $f_{NN} : [p] \to [p]$ such that $\|\mathbf{x_j} - \mathbf{x_{f_{NN}(j)}}\|_\mathbf{2} = \min_{\mathbf{y} \in \mathbf{S}} \|\mathbf{x_j} - \mathbf{y}\|_\mathbf{2}$, $\forall \mathbf{j} \in [\mathbf{p}]$

Recall from chapter 1 that the complexity is $\mathcal{O}(p^2 D)$ for the linear scan solution, shown in 17. In this (exact) solution method, we compute all pairwise distances $\|\mathbf{x_i} - \mathbf{x_j}\|_\mathbf{2}$, $\forall \mathbf{i}, \mathbf{j} \in [\mathbf{p}]$.

However, in many applications either p or D can be large, which means the exact computation using this straightforward method becomes computationally intractable. We will sacrifice accuracy in order to achieve faster results. Now consider the following variant of the nearest neighbor problem.

3.7.2 A Modified Problem on Our Way to a Fast Approximate Nearest Neighbor Algorithm

Definition 3.7.4 ((c,r)-Nearest Neighbor Problem)**.** *(c,r)-NN problem: Given $S = \{\mathbf{x_0}, \ldots, \mathbf{x_{p-1}}\} \subseteq \mathbb{R}^D$ find $f : [p] \to [p] \cup \{-1\}$ so that both:*

1. $d(\mathbf{x_j}, \mathbf{x_{f(j)}}) \leq \mathbf{cr}$, $\forall \mathbf{j} \in [\mathbf{p}]$ *such that* $\exists i \in [p]$ *with* $d(\mathbf{x_j}, \mathbf{x_i}) \leq \mathbf{r}$.

   The idea is that if there is a point that is r-close to $\mathbf{x_j}$, then the assignment function will return a point that is almost as close. The possible error in the nearest neighbor to $\mathbf{x_j}$ is quantified by c.

2. $f(j) = -1$ *if* $\nexists i \in [p]$ *with* $d(\mathbf{x_j}, \mathbf{x_i}) \leq \mathbf{cr}$.

   In other words, if there is no point that is cr-close to the query point $\mathbf{x_j}$, then the assignment function will indicate this fact by returning -1.

The diagram shows schematically how this new problem simplifies nearest neighbor. The query point in the diagram is $\mathbf{x_j}$. If $\mathbf{x_k}$ is within a distance r of the query point, then our assignment function can return either $\mathbf{x_k}$ or $\mathbf{x_i}$. If there are no points within cr distance to the query (i.e. remove points $\mathbf{x_i}$ and $\mathbf{x_k}$) then the function returns -1. If there is a point within cr but no point within r (i.e. remove only $\mathbf{x_k}$) then there is no requirement that the function assign any particular value to the nearest neighbor (e.g. returning $\mathbf{x_i}$ or -1 are possible)

Note that for this, and most other examples, we will concern ourselves with the Euclidean distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\mathbf{2}$.

Goal. *As a floor to solution run-time, we should at least read in all the data, which takes $\Omega(pD)$-time.*

To understand this lower bound on run-time, consider what might happen by withholding points from consideration in a solution. There is no way then to guarantee that the withheld points are not nearest neighbors to a given query point in this scenario. A pause to recall some

Definition 3.7.5 ($\mathcal{O}, \Omega, \Theta$ complexity)**.** *1. Let $g : \mathbb{R} \to \mathbb{R}^+$. We say g is $\mathcal{O}(h)$, $h : \mathbb{R} \to \mathbb{R}^+$, if $\exists C, x_0 \in \mathbb{R}$ such for all $y > x_0$ $g(y) \leq Ch(y)$.*

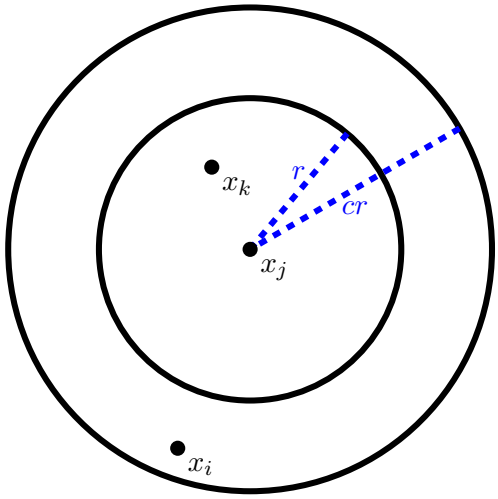


Figure 3.1: (c, r)-NN with query point x_j

2. *Let* $g : \mathbb{R} \to \mathbb{R}^+$. *We say* g *is* $\Omega(\tilde{h})$, $\tilde{h} : \mathbb{R} \to \mathbb{R}^+$, *if* $\exists C, x_0 \in \mathbb{R}$ *such for all* $y > x_0$ $g(y) \geq C\tilde{h}(y)$.

3. *If* g *is* $\mathcal{O}(h)$ *and* $\Omega(h)$ *then* g *is* $\Theta(h)$.

In order to achieve our goal of improving on nearest neighbor beyond $\mathcal{O}(p^2 D)$, we will take our set $S \subset \mathbb{R}^D$ and project each of the points onto a random vector and find nearest neighbors of the projections which are lower dimensional. Schematically,

Where the procedure is to generate a random line in the direction of $\mathbf{g}$, and then

1. Project all points of S onto $\mathbf{g}$, i.e. calculate $\langle \mathbf{g}, \mathbf{x_1} \rangle, \langle \mathbf{g}, \mathbf{x_2} \rangle, \ldots$

2. Sort the distances $\{\langle \mathbf{g}, \mathbf{x_j} \rangle\}_{j=1}^{4}$

3. Read off nearest neighbors from the sorted list

So, using the schematic our DEFINE ASSIGNMENT FUNCTION assignment function could be

$$f(1) = 2, f(2) = 1, f(3) = 4, f(4) = 2$$

Now let's consider complexity. The inner product of a point with a random vector takes on order D operations and must be performed for all points, so step one takes on order PD-time. Sorting lists is a well understood problem in computer science and can be accomplished on order $P \log D$ time. Finally, scanning the list for a nearest neighbor takes P time, so overall our complexity is $\mathcal{O}(PD + P \log D + P)$.

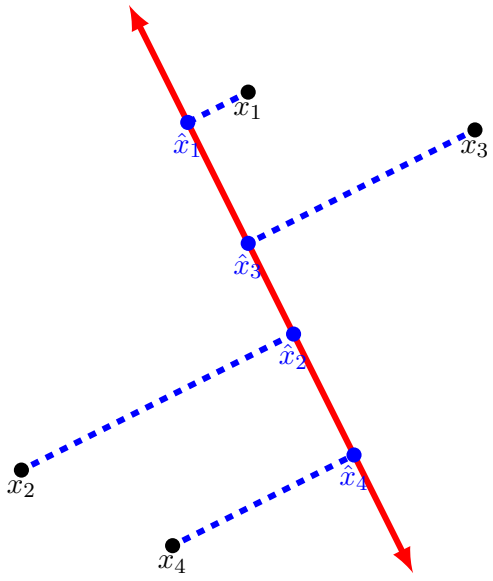


Figure 3.2: (c, r)-NN using 2D data set

Definition 3.7.6 (Locality Sensitive Hash Function)**.** *We call a random function* $h : \mathbb{R}^D \to \mathbb{Z}$ *a Locality Sensitive Hash function if* $\exists p_1, p_2 \in (0,1), p_1 > p_2$*, so that the following properties hold for any two fixed points* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathbf{D}}$*:*

1. *If* $\|\mathbf{x} - \mathbf{y}\| < \mathbf{r}$ *then* $h(\mathbf{x}) = \mathbf{h}(\mathbf{y})$ *with at least probability* p_1
2. *If* $\|\mathbf{x} - \mathbf{y}\| > \mathbf{cr}$ *then* $h(\mathbf{x}) = \mathbf{h}(\mathbf{y})$ *with probability at most* p_2

So a LSH function will hash similar points to the same integer and points which are dissimilar to different integers. We now consider a particular example of such a function.

Example 3.7.7. *Fix two points* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathbf{D}}$*. We define a hash function* $h : \mathbb{R}^D \to \mathbb{Z}$ *as follows*

$$h(\mathbf{x}) = \left\lfloor \frac{\langle \mathbf{g}, \mathbf{x} \rangle + \mathbf{u}}{\mathbf{w}} \right\rfloor$$

where $\mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ *and* $u \sim$ *Uniform*$(0, w)$ *and* w *is a fixed positive number.*

In the following then, we regard h to be a function of the two random variables of u and $\mathbf{g}$ where the points themselves $\mathbf{x}, \mathbf{y}$ are fixed. The key questions then to consider are what are p_1 and p_2 in 3.7.7?

By FIX BROKEN LINK Lemma **??**, since the event $h(\mathbf{x}) = \mathbf{h}(\mathbf{y})$ implies $|\langle \mathbf{g}, \mathbf{x} \rangle - \langle \mathbf{g}, \mathbf{y} \rangle| < w$, we have

$$P_{u,\mathbf{g}}\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y})\right] = P\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y}) \cap |\langle \mathbf{g}, \mathbf{x} \rangle - \langle \mathbf{g}, \mathbf{y} \rangle| < \mathbf{w}\right]$$

and so applying Lemma 3.5.6 we have the following equivalent expression as an integral

$$P\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y}) \cap \lfloor \langle \mathbf{g}, \mathbf{x} \rangle - \mathbf{g}, \mathbf{y} \rfloor < \mathbf{w}\right] =$$
$$\int_0^w P\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y}) || \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle| = \mathbf{z}\right] P\left[|\langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle| = z\right] dz$$

Let's consider each of the probabilities that appear in the integrand,

- The probability below is a probability of only the random variable u since all other quantities are fixed.
$$P\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y}) || \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle| = \mathbf{z}\right]$$
So, given a particular $\mathbf{x}, \mathbf{y}$ and $z \in [0, w]$ such that $|\langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle| = z$ we need to determine the probability that an offset u results in $\langle \mathbf{g}, \mathbf{x} \rangle$ and $\langle \mathbf{g}, \mathbf{y} \rangle$ falling into the same "bin" of width w, i.e. they are hashed to same integer $h(\mathbf{x})$. IMPROVE THIS DESCRIPTION Without loss of generality, suppose $a = \langle \mathbf{g}, \mathbf{x} \rangle - \mathbf{wh}(\mathbf{x}) < \langle \mathbf{g}, \mathbf{y} \rangle - \mathbf{wh}(\mathbf{x}) = \mathbf{b}$. So, what then is the probability that $h(\mathbf{x}) = \mathbf{h}(\mathbf{y})$ as a function of u, given z? This reduces to considering which offsets results in a segment of length z being contained entirely in a segment of length w - and due to the periodic nature of moving bin boundaries, it suffices to consider the case when $a = 0$. To see why this is, the diagram shows three possible scenarios for an offset u.

  When $u = 0$, the bin boundary of bin 0 is aligned with a. As we imagine u taking values from 0 to w we see the bin boundaries take all possible locations before ending back in a position where a is again exactly aligned with a bin boundary (now the bin corresponding to 1). So, for a non-zero a the starting condition is different, but the overall "movie" is the same. We need then consider for what proportion of "frames" for this movie the segment of length z is entirely contained in a single bin. If the movie is of length w then for the first z frames, the segment is split between two bins, i.e. with probability $\frac{z}{w}$ the segment is split between bins and $h(\mathbf{x}) \neq \mathbf{h}(\mathbf{y})$. The complementary event then $1 - \frac{z}{w} = \frac{w-z}{w}$ is the probability that the segment is contained in a single bin, and thus $h(\mathbf{x}) = \mathbf{h}(\mathbf{y})$

- We now consider
$$P\left[|\langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle| = z\right]$$
We know from Lemma 3.5.7 and subsequent discussion that $\langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle \sim \mathcal{N}(\mathbf{0}, \|\mathbf{x} - \mathbf{y}\|_\mathbf{2}^\mathbf{2})$. Accounting for absolute values then, the probability is given by
$$\frac{\sqrt{2}}{\|\mathbf{x} - \mathbf{y}\|_\mathbf{2}\sqrt{\pi}} \exp\left(-\frac{z^2}{2\|\mathbf{x} - \mathbf{y}\|_\mathbf{2}^\mathbf{2}}\right)$$

Combining the results then above, conducting a change of variables and an integration by parts, we have for $n = \|\mathbf{x} - \mathbf{y}\|_\mathbf{2}$ then an expression which computes the probability that

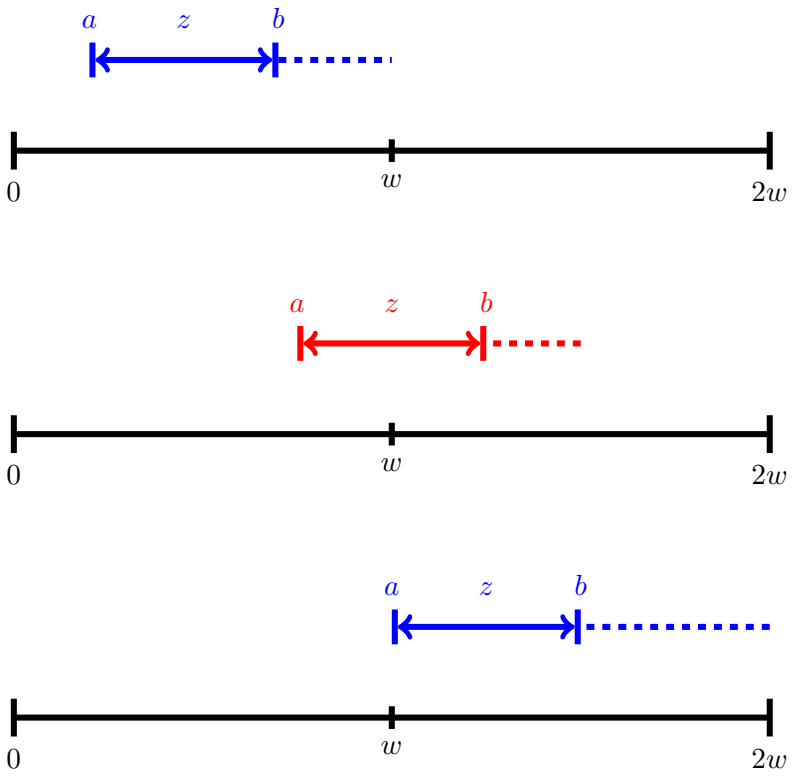


Figure 3.3: We can compute $P[h(x) = h(y) \mid |\langle g, x - y\rangle| = z]$ by considering the range of intervals with length z contained in only one bin of size w.

two points hash to the same integer as a function of the distance between the points:

$$\begin{aligned} p_w(n) &= \int_0^w P\left[h(\mathbf{x}) = \mathbf{h}(\mathbf{y}) || \langle \mathbf{g}, \mathbf{x} - \mathbf{y}\rangle| = \mathbf{z}\right] P\left[|\langle \mathbf{g}, \mathbf{x} - \mathbf{y}\rangle| = z\right] dz \\ &= \frac{2}{\sqrt{\pi}} \int_0^{\frac{w}{n\sqrt{2}}} e^{-z^2} dz + \sqrt{\frac{2}{\pi}} \frac{n}{w} \left[e^{-\left(\frac{w}{n\sqrt{2}}\right)^2} - 1\right] \end{aligned}$$

Taking the derivative with respect to n we have

$$\begin{aligned}\frac{d}{dn}p_w(n) &= \frac{d}{dn}\left[\frac{2}{\sqrt{\pi}}\int_0^{\frac{w}{n\sqrt{2}}} e^{-z^2}dz + \sqrt{\frac{2}{\pi}}\frac{n}{w}\left[e^{-\left(\frac{w}{n\sqrt{2}}\right)^2}-1\right]\right] \\ &= -\frac{2}{\sqrt{\pi}}\frac{w}{\sqrt{2}n^2}e^{\left(\frac{w}{\sqrt{2}n}\right)^2} + \sqrt{\frac{2}{\pi}}\frac{1}{w}\left[e^{-\left(\frac{w}{n\sqrt{2}}\right)^2}-1\right] + \sqrt{\frac{2}{\pi}}\frac{n}{2}\frac{2w^2}{2n^3}\left[e^{-\left(\frac{w}{n\sqrt{2}}\right)^2}\right] \\ &= \sqrt{\frac{2}{\pi}}\frac{1}{w}\left[e^{-\left(\frac{w}{n\sqrt{2}}\right)^2}-1\right]\end{aligned}$$

We can see that $\frac{d}{dn}p_w(n) < 0$ which is to say that the function is monotonically decreasing, thus when $n = \|\mathbf{x}-\mathbf{y}\|_\mathbf{2} < \mathbf{r}$ we have that $p_w(n) > p_w(r)$. That is $p_w(r) = p_1$ from Definition 3.7.6. Furthermore, when for $c > 1$ we have $n = \|\mathbf{x}-\mathbf{y}\|_\mathbf{2} > \mathbf{cr}$ we know that $p_w(cr) > p_w(n)$. That is $p_w(cr) = p_2$ from Definition 3.7.6.

We have shown quantitatively that the probability of hashing to the same integer is greater if the points are close and smaller if they are farther away. The following Lemma summarizes then what we have demonstrated.

Lemma 3.7.8. *Let* $\mathbf{g} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \mathbf{u} \sim ([\mathbf{0},\mathbf{w}])$ *and* $w \in \mathbb{R}^+$. *Then* $h(\mathbf{x}) = \left\lfloor \frac{\langle \mathbf{g},\mathbf{x}\rangle + \mathbf{u}}{\mathbf{w}} \right\rfloor$ *is a LSH function* $\forall r \in \mathbb{R}^+$ *and* $c \in (0,1)$ *with respect to Euclidean distance. It has* $p_1 = p_w(r) > p_2(cr) = p_2$ *where*

$$p_w(n) = erf\left(\frac{w}{\sqrt{2}n}\right) + \sqrt{\frac{2}{\pi}}\frac{n}{w}\left[e^{-\left(\frac{w}{\sqrt{2}n}\right)^2}-1\right]$$

Define the error function way above.

At present, our hashing function only addresses what happens for two fixed vectors $\mathbf{x}$ and $\mathbf{y}$. How can we use this function to build a hashing scheme which works for a large data set with possibly billions of data points? We will accomplish this by repeating the hash using independent draws of the random variables and using the collection of these to hash the entire set.

Definition 3.7.9. *Let* $g_k : S \to \mathbb{Z}^k$ *be a new LSH function created by using* k *i.i.d LSH functions of the type in Example 3.7.7, i.e.* $h_1,\ldots,h_k$, $g_k(\mathbf{x}) = (\mathbf{h_1}(\mathbf{x}),\ldots,\mathbf{h_k}(\mathbf{x}))$

Definition 3.7.10. *When* $g_k : S \to \mathbb{Z}^k$ *has the properties, for some fixed* $\mathbf{x} \in \mathbf{S}$

1. *If for* $\mathbf{y} \in \mathbf{S}$ *where* $d(\mathbf{x},\mathbf{y}) \geq \mathbf{rc}$ *then* $g_k(\mathbf{x}) \neq \mathbf{g_k}(\mathbf{y})$.
2. *If for at least one* $\mathbf{y} \in \mathbf{S}$ *it happens that* $d(\mathbf{x},\mathbf{y}) \leq \mathbf{r}$ *then* $g_k(\mathbf{x}) = \mathbf{g_k}(\mathbf{y})$.

then it is a satisfactory LSH function for $\mathbf{x} \in \mathbf{S}$.

In other words, g_k does not hash a dissimilar points to the same thing as the query and also if there is some point close to our query, then g_k should hash a close point and query to the same thing. Note that we now have a more expansive criteria for our hash than in Example 3.7.7; it requires something hold true for all other points in the set, not simply a pair.

Definition 3.7.11. *For* $\mathbf{x} \in \mathbf{S}$ *we denote the nearest neighbor of* $\mathbf{x}$ *as*

$$\mathbf{x}^* = \arg\min_{\substack{\mathbf{y}\in\mathbf{S} \\ \mathbf{x}\neq\mathbf{y}}} \mathbf{d}(\mathbf{x},\mathbf{y})$$

We wish to know bounds on the probability that g_k fails to meet each of the properties in Definition 3.7.10. Consider the first property, it fails when there exists a point which is far away that nevertheless hashes to the same vector as $\mathbf{x}$. In order to hash to the same vector in $\mathbb{Z}^k$ all k component hashes need to incorrectly hash "far" points to the same integer. For each component that happens with probability p_2. Thus:

$$\begin{aligned} P\left[g_k \text{ fails 1 }\right] &= P\left[\exists \mathbf{y} \in \mathbf{S} \text{ s.t. } \mathbf{g_k}(\mathbf{x}) = \mathbf{g_k}(\mathbf{y}),\, \mathbf{d}(\mathbf{x},\mathbf{y}) \geq \mathbf{rc}\right] \\ &\leq |S| P\left[g_k(\mathbf{x}) = \mathbf{g_k}(\mathbf{y}),\, \mathbf{d}(\mathbf{x},\mathbf{y}) \geq \mathbf{rc}\right] \\ &\text{Add more detail to this calculation} \\ &\leq |S| p_2^k \end{aligned}$$

where we have used a union bound.

How can we bound the probability that the second property in Definition 3.7.9 fails? The complementary event is that all points which are close hash to the same vector in $\mathbb{Z}^k$. This means that component-wise the k hashes all need to hash $\mathbf{x}$ and $\mathbf{y}$ to the same integer, which happens with probability p_1 for each component. Thus:

$$\begin{aligned} P\left[g_k \text{fails 2}\right] &= P\left[\exists \mathbf{y} \in \mathbf{S}\mathbf{g_k} \text{ s.t. } (\mathbf{x}) \neq \mathbf{g_k}(\mathbf{y}),\, \mathbf{d}(\mathbf{x},\mathbf{y}) < \mathbf{r}\right] \\ &\leq 1 - |S| P\left[g_k(\mathbf{x}) = \mathbf{g_k}(\mathbf{y}),\, \mathbf{d}(\mathbf{x},\mathbf{y}) \geq \mathbf{rc}\right] \\ &\leq 1 - p_1^k \end{aligned}$$

The probability that g_k is a satisfactory LSH then can be bounded from below by

$$\begin{aligned} P\left[g_k \text{satisfies 1 and 2}\right] &= 1 - P\left[g_k \text{fails 1 or 2}\right] \\ &\geq 1 - P\left[g_k \text{fails 1}\right] - P\left[g_k \text{fails 2}\right] \qquad \geq 1 - |S| p_2^k - (1 - p_1^k) \\ &= p_1^k \left(1 - |S| \left(\frac{p_2}{p_1}\right)^k\right) \end{aligned}$$

Let $k = \log_{p_1/p_2}(2|S|)$ and $\rho = \frac{\log p_1}{\log p_2}$, then this simplifies as follows:

$$\begin{aligned}
p_1^k \left(1 - |S| \left(\frac{p_2}{p_1}\right)^k\right) &= p_1^{\left(\log_{p_1/p_2}(2|S|)\right)} \left(1 - |S| \left(\frac{p_2}{p_1}\right)^{\left(\log_{p_1/p_2}(2|S|)\right)}\right) \\
&= p_1^{\left(\log_{p_1/p_2}(2|S|)\right)} \left(1 - |S| \left(\frac{p_1}{p_2}\right)^{\left(\log_{p_1/p_2}\left(\frac{1}{2|S|}\right)\right)}\right) \\
&= p_1^{\left(\log_{p_1/p_2}(2|S|)\right)} \left(1 - |S| \left(\frac{1}{2|S|}\right)\right) \\
&= \frac{1}{2} p_1^{\left(\log_{p_1/p_2} 2|S|\right)} \\
&= \frac{1}{2} \left[p_1^{\left(\frac{\log_{p_1}(2|S|)}{\log_{p_1}\left(\frac{p_1}{p_2}\right)}\right)}\right] \\
&= \frac{1}{2} \left[p_1^{\left(\log_{p_1}(2|S|)\right)}\right]^{\frac{1}{\frac{\log\left(\frac{p_1}{p_2}\right)}{\log p_1}}} \\
&= \frac{1}{2} \left[2|S|\right]^{\frac{\frac{\log p_1}{\log p_2}}{\frac{\log p_1}{\log p_2} - 1}} \\
&= \frac{1}{2} \left[2|S|\right]^{\frac{\rho}{1-\rho}}
\end{aligned}$$

We gather then this in the following Lemma

Lemma 3.7.12. *If $k = \log_{p_1/p_2}(2|S|)$ and $\rho = \frac{\log p_1}{\log p_2}$ then g_k as in Definition 3.7.9 will be a satisfactory LSH as described in Definition 3.7.10 for any given $\mathbf{x} \in \mathbf{S}$ with probability at least*

$$\frac{1}{2} \left[2|S|\right]^{\frac{\rho}{1-\rho}}$$

Lemma 3.7.13. *If we generate $L \geq 2(2|S|)^{\frac{\rho}{1-\rho}} \log\left(\frac{|S|}{1-\sigma}\right)$ i.i.d. hash functions $g_k^i : S \to \mathbb{Z}^k, j = 1, \ldots, L$ with $k = \log_{p_1/p_2}(2|S|)$, $\rho = \frac{\log p_1}{\log p_2}$ then the following will hold with probability at least σ:*

$\forall x \in S$, $\exists \ell \in [L]$ *s.t.* g_k^ℓ *is a satisfactory LSH in the sense of Definition 3.7.10 for* x

Proof. Let $\delta = \frac{1}{2}\left(\frac{1}{2|S|}\right)^{\frac{\rho}{1-\rho}}$ and fix $x \in S$. The probability that a g_k^i will be unsatisfactory for x is at most $(1 - \delta)$ by Lemma 3.7.12. Thus the probability that all g_k^i will fail to be

satisfactory is at most $(1-\delta)^L$. We can use Fact 3.3.13 to conclude that

$$(1-\delta)^L \leq e^{-\delta L} \leq e^{\delta 2(2|S|)^{\frac{\rho}{1-\rho}} \log\left(\frac{|S|}{1-\sigma}\right)} = e^{\log\left(\frac{1-\sigma}{|S|}\right)} = \frac{1-\sigma}{|S|}$$

So the probability that for every x all hash functions fail to be satisfactory is bounded by the union of $|S|$ such probabilities seen above, i.e. $1-\sigma$. The complementary event, that for every x at least one hash doesn't fail, is then at least σ. □

We now have the results needed to construct a randomized algorithm that solves the (c,r)-NN problem.

SIMPLY THIS CODE – for each x, compute it's nearest neighbor from things hashed to the same bucket as it in all the L LSH functions

Algorithm 18 LSH for (c,r)-NN

Input: $S \subseteq \mathbb{R}^D$, $d(x,y) = \|\mathbf{x}-\mathbf{y}\|_\mathbf{2}$
Output: $f : S \to S \cup \{\infty\}$ a (c,r)-NN map

for $\mathbf{x}$ in S **do**
  $\forall \ell \in L$ compute $g_k^\ell(\mathbf{x})$
end for
$\forall \mathbf{x} \in \mathbf{S}, \mathbf{f}(\mathbf{x}) = (\infty, \ldots, \infty)$
for each g_k^ℓ, $\ell \in [L]$ **do**
  for each n in $g_k^\ell(S)$ where $\left|(g_k^\ell)^{-1}(n)\right| \geq 2$ **do**
    for each $\mathbf{x}$ in $\left(g_k^\ell\right)^{-1}(n)$ **do**
      choose exactly one $\mathbf{y}$ in $\left(g_k^\ell\right)^{-1}(n) \setminus \{x\}$
      if $\|\mathbf{x}-\mathbf{y}\|_\mathbf{2} < \min\{\|\mathbf{x}-\mathbf{f}(\mathbf{x})\|_\mathbf{2}, \mathbf{cr}\}$ **then**
        $f(\mathbf{x}) \leftarrow \mathbf{y}$
      end if
    end for
  end for
end for

The first for loop has $\mathcal{O}(DKL|S|)$ run-time, since each element of S must be projected with an inner-product of length D, K-times for each of the L hash functions.

Next we move onto the bottom block, after the first end for. We analyze the run-time from the inside out: the inner if statement requires comparison of norms and so will require $\mathcal{O}(D)$ comparisons. The two most inner for loops could in the worst case scenario iterate over $|S|$ vectors, each of which needs to compared on K entries (in the case that all vectors hash to the same vector $\mathbf{n}$) and therefore overall the inner loop has worst case run-time complexity $\mathcal{O}(|S|DK)$. The outer-loop iterates L times, and so overall the entire loop has at most $\mathcal{O}(DKL|S|)$

Recall that algorithm 17 finds exact answers in $\mathcal{O}(D|S|^2)$ (which would naturally solve the (c,r)-NN problem as well). So for what types of problems does this represent a cost-savings? From our Lemma 3.7.13 and analysis we have that if

$$K \geq \log_{p_1/p_2}(2|S|)$$
$$L \geq 2\,(2|S|)^{\frac{\rho}{1-\rho}} \log\left(\frac{|S|}{1-\sigma}\right)$$

then with probability at least σ our algorithm should produce a satisfactory solution to the (c,r)-nearest neighbor problem. That is for complexity

$$\mathcal{O}\left(D|S|^{1+\frac{\rho}{1-\rho}} \log\left(\frac{|S|}{1-\sigma}\right) \log_{p_1/p_2}(2|S|)\right)$$

and if we fix $w = 3r$ and $c = 3$ in the Definition of the component hash functions h then $\frac{\rho}{1-\rho} \approx 0.449$ and $p1/p2 \geq 1.99$ so in this scenario we have saved something on the order of $|S|^{1/2}$ from the naive solution, which represents a significant savings for large sets.

Theorem 3.7.14. *Choose $\sigma \in (0,1)$, $S \subset \mathbb{R}^D$. Then $\forall r > 0$, $(3,r)$-NN problem can be solved for S with respect to Euclidean distance with probability σ in time*

$$\mathcal{O}\left(D|S|^{1.5} \log\left(\frac{|S|}{1-\sigma}\right) \log_{1.99}(2|S|)\right)$$

Note.

1. *Having $\rho = \frac{\log p_1}{\log p_2}$ small is crucial. The following result is described in [16]:*

   $\forall q \in (0,2]$ and $r \in \mathbb{R}^+, \delta.c \in (1,\infty)$ $\exists$ an LSH function $h : \mathbb{R}^D \to \mathbb{Z}$ with respect to $\|\cdot\|_q$ having $\rho \leq \delta \max\left(c^{-q}, c^{-1}\right)$

2. *In [3] we have the following result which shows a near optimal result for Euclidean distance: $\exists$ a LSH with respect to $\|\cdot\|_2$, $\forall r \in \mathbb{R}^+$,$c \in (1,\infty)$ that has*

   $$\rho = \frac{1}{c^2} + \mathcal{O}\left(\frac{\log\log|S|}{\log^{1/3}|S|}\right)$$

   for any given $S \subset \mathbb{R}^D$

3. *In [39] we have a lower-bound on ρ. It states that for large D there exists an r and $p_2 \geq 2^{-\mathcal{O}(D)}$ for which $\rho \geq \frac{0.462}{c^q}$ for any LSH with respect to $\|\cdot\|_q$, $\forall c, q \geq 1$.*

More detail on Cauchy random variable stuff.... Move order around.

Exercise 3.7.1. *Use the Definition of Cauchy random variables and discussion below to prove that $h : \mathbb{R}^D \to \mathbb{Z}$ in 3.7.7 is still a Locality Hashing Function $\forall w, r \in \mathbb{R}^+$ and $c \in (1,\infty)$ with respect to $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_1$ when each entry of $\mathbf{g}$ is an i.i.d Cauchy random variable with density $f_{0,1}(x)$ from 3.7.15.*

Definition 3.7.15 (Cauchy Random Variable)**.** *A Cauchy random variable is real value* $X \sim Cauchy(x_0, \gamma_0)$ *that has the probability density function*

$$f_{x_0,\gamma}(x) = \frac{1}{\pi\gamma\left(1+\left(\frac{x-x_0}{\gamma}\right)^2\right)} = \frac{1}{\pi}\left(\frac{\gamma}{(x-x_0)^2+\gamma^2}\right)$$

So for example when $x_0 = 0, \gamma = 1$, $f_{0,1}(x) = \frac{1}{\pi(1+x^2)}$

Interestingly, the mean and variance of Cauchy random variable are undefined, as can be seen by writing the associated integrals.

Consider two independent Cauchy random variables $X \sim Cauchy(x_0, \gamma_0)$ *and* $Y \sim Cauchy(y_0, \delta_0)$*. Then*

1. $kX + L \sim Cauchy(kx_0 + L, |k|\gamma_0)$
2. $X + Y \sim Cauchy(x_0 + y_0, \gamma_0 + \delta_0)$

These two properties constitute the stable distribution property.

Exercise 3.7.2. *For* $c = 3$ *and* $w = 3r$ *verify that* $\frac{\rho}{1-\rho} \approx 0.449$ *and* $\frac{p_1}{p_2} \geq 1.99$*. Can you improve for any arbitrary* r*?*

Exercise 3.7.3. *Let*

$$R := \frac{\min_{\mathbf{x}\in\mathbf{S}} \|\mathbf{x}^* - \mathbf{x}\|_\mathbf{2}}{\max_{\mathbf{x}\in\mathbf{S}} \|\mathbf{x}\|_\mathbf{2}} < 1$$

choose $\sigma \in (0,1)$ *such that* , $\frac{\sigma}{\log_{4/3}(R)} < 1$*. Prove that we can solve a sequence of* $(3, r)$*-NN problems to get* $f_{NN}^A : S \to S$ *satisfying*

$$\|f_{NN}^A(\mathbf{x}) - \mathbf{x}\|_\mathbf{2} \leq \mathbf{4}\|\mathbf{x} - \mathbf{x}^*\|_\mathbf{2}$$

for all $\mathbf{x}^* \in \mathbf{S}$ *with probability at least* σ *in Link back to Def of* x^**, and define what* R *is!*

$$\mathcal{O}\left(D|S|^{1.5}\log\left(\frac{|S|\log_{4/3}\left(\frac{1}{R}\right)}{1-\sigma}\right)\log_{1.99}(|S|)\log_{1.99}\left(\frac{1}{R}\right)\right)$$

3.8 Hashing (CMSE 890 Lecture 6)

Definition 3.8.1 (Perfect Hash)**.** *A random function* $h : U \to [0,1]$ *with the properties that* $\forall a \in U$

1. $h(a)$ *is a uniform random variable in* $[0,1]$
2. $h(a)$ *is independent of* $h(b)$ *for* $a \neq b$

There are several ways in which we could generate a Perfect simple i.i.d uniform hash function:

1 When $a \in [M]$ Connect M to cardinality of universe U below!, we generate a random variable $X_a \in [0,1]$ and store (a, X_a).
When b is entered and we want to hash it:

- Check and see if we already generated a random value for b
- If we have already generated a random number for b, X_b, return it
- Else generate a new random number for b and store (b, X_b) in the list.

This approach require $\Omega(M)-$memory and $\mathcal{O}(\log M)$ evaluation-time per hash.

2 Generate M i.i.d uniform random variable.
When $a \in [M]$ arrives, output the a^{th} entry in my random list.
This approach requires $\Omega(M)-$memory and $\mathcal{O}(1)$ hash time.

Note. *Storing arrays of M uniformly generated random numbers to use as hashes is a problem if $M \gg 1$. We may also object to this approach on the grounds that reusing or having foreknowledge about the array would torpedo the randomness aspect that we want. An adversary given access to the hash could effect any count or miscount he wanted by choosing the sequence of inputs (or simply by re-ordering them).*

3.8.1 Hashing in Practice(Partially breaking the independence assumption)

In practice, we consider the following procedure to generate a usable and tractable hash:

1. Select a very large prime number $P > M$
2. Choose two random, independent uniformly distributed integer values $a, b \in [P]$
3. Compute for $x \in [P]$ hash to the value $h_{a,b}(x)$ in the following way

$$h_{a,b}(x) = \frac{(ax+b) \mod P}{P} \in [0,1)$$

Lemma 3.8.2.

$h_{a,b}(x) = \dfrac{(ax+b) \mod P}{P}$ *is uniformly distributed in* $\{0, \frac{1}{P}, \cdots, \frac{P-1}{P}\} \subset [0,1]$ *for* $x \in [P]$.

Proof. Let $x, j \in [P]$. Then :

$$\begin{aligned}\mathbb{P}\left[h_{a,b}(x) = \frac{j}{P}\right] &= \mathbb{P}[ax+b = j \mod P] \\ &= \mathbb{P}[ax = (j-b) \mod P]\end{aligned}$$

If $x \neq 0$, $\mathbb{P}[a = (x^{-1})(j-b) \mod P] = \frac{1}{P}$ since a is uniformly distributed in $[P]$
Else if $x = 0$, $\mathbb{P}[b = j \mod P] = \frac{1}{P}$ since b is uniformly distributed in $[P]$

□

Lemma 3.8.3. *Let $x, y \in [P]$, $x \neq y$. Then:*

$$\mathbb{P}\left[h_{a,b}(x) = \frac{j}{P} \text{and } h_{a,b}(y) = \frac{l}{P}\right] = \frac{1}{P^2} \qquad \forall j, l \in [P].$$

Thus $h_{a,b}(x)$ and $h_{a,b}(y)$ are pairwise independent.

Proof. Assume without loss of generality $x \neq 0$, thus $\exists x^{-1} \in \mathbb{Z}_P$

$$\begin{aligned}
\mathbb{P}\left[h_{a,b}(x) = \frac{j}{P}, h_{a,b}(y) = \frac{\ell}{P}\right] &= \mathbb{P}\left[a = x^{-1}(j-b) \mod P, b = \ell - ay \mod P,\right] \\
&= \mathbb{P}\left[a = x^{-1}(j-\ell+ay) \mod P, b = \ell - x^{-1}(j-b)y \mod P,\right] \\
&= \mathbb{P}\left[a = x^{-1}(j-\ell)(1-x^{-1}y)^{-1} \mod P, b = (\ell - x^{-1}jy)(1-x^{-1})^{-1} \mod P,\right] \\
&= \frac{1}{P^2}
\end{aligned}$$

Thus $h_{a,b}(x)$ and $h_{a,b}(y)$ are pairwise independent. □

Observe that the hash function described in this section, and the resulting random variables of the type $h_{a,b}(x_1), \ldots, h_{a,b}(x_n)$ have the product property (definition **??**) only for pairs. We have that $h_{a,b}(x), h_{a,b}(y)$ for $0 \neq x, y, z$ are not independent.
In fact if we know $h_{a,b}(x)$ and $h_{a,b}(y)$ then $h_{a,b}(z)$ is deterministically dependent.
To see why the hash is not three-wise independent, consider

$$\mathbb{P}\left[h_{a,b}(x) = j/P, h_{a,b}(y) = \ell/P, h_{a,b}(z) = f/P\right].$$

From the proof of Lemma 3.8.3 we have that:

$$\begin{aligned}
a &= x^{-1}(j-\ell)(1-x^{-1}y)^{-1} \mod P \\
b &= (\ell - x^{-1}yj)(1-x^{-1}y)^{-1} \mod P
\end{aligned}$$

So then

$$h_{a,b}(z) = \frac{az+b \mod P}{P} = \frac{(1-x^{-1}y)^{-1}\left[(j-\ell)z + (\ell - x^{-1}yj)\right]}{P} = f_{x,y,z,\ell,j}$$

i.e. once we've selected values for j and ℓ there is only one possible value that $h_{a,b}(z)$ can hash to (it is completely determined by those two values). So

$$\mathbb{P}\left[h_{a,b}(x) = j/P, h_{a,b}(y) = \ell/P, h_{a,b}(z) = f/P\right] = \begin{cases} \frac{1}{P^2} & \text{if } f = (1-x^{-1}y)^{-1}\left[(j-\ell)z + (\ell - x^{-1}yj)\right] \\ 0 & \text{otherwise} \end{cases}$$

If the values were in fact independent, we would need that the probability for any particular three values as $1/P^3$.
Hash functions that have k-wise independence exist, however are beyond the scope of this course and will require the use of polynomials over some finite fields.
We conclude our discussion of this hash function with a consideration for how large the prime P should be. In the case of a perfect hash h we know that $\mathbb{P}\left[h(x) = h(y)\right] = 0$ when $x \neq y$. With our imperfect hash $h_{a,b}$, we would like to have $h_{a,b}(x) = h_{a,b}(y)$ for $x \neq y \in [M]$ with high probability $1 - q$ for $q \in [0, 1)$. Now, for $x \neq y$:

$$\begin{aligned}\mathbb{P}\left[h_{a,b}(x) = h_{a,b}(y)\right] &= \sum_{j=0}^{P-1} \mathbb{P}\left[h_{a,b}(x) = \frac{j}{P}, h_{a,b}(y) = \frac{j}{P}\right] \\ &= \sum_{j=0}^{P-1} \frac{1}{P^2} \\ &= \frac{1}{P}.\end{aligned}$$

Thus by the union bound, we have:

$$\mathbb{P}\left[\exists x, y \in [M] : h_{a,b}(x) = h_{a,b}(y)\right] \leq \binom{M}{2}\frac{1}{P} = \frac{M(M-1)}{2P}.$$

$\rightsquigarrow$ If $P \geq \frac{M(M-1)}{2q}$ for some $q \in [0, 1)$ we get our desired result i.e

$$\begin{aligned}\mathbb{P}\left[\nexists x, y \in [M] : h_{a,b}(x) = h_{a,b}(y)\right] &= 1 - \mathbb{P}\left[\exists x, y \in [M] : h_{a,b}(x) = h_{a,b}(y)\right] \\ &\geq 1 - q.\end{aligned}$$

$\rightsquigarrow$ $P \approx M^3$ is the rule of thumb.

3.8.2 Problem Statement and Naive Solution

Another problem which is similar to counting objects is the distinct elements problem. Here we concerned with determining the number of distinct elements which appear in a given sequence, as opposed to the frequency of a particular element.

Formally, given a sequence $\{z_j\}_{j=1}^N$ where $\forall j$, $z_j \in U$, and $|U| = D$ we wish to compute the cardinality of $\{z_j\}_{j=1}^N$ as a set.

Goal. *Estimate the number of distinct elements in a sequence using a number of bits independent of both N and D.*

One can imagine many different settings where such a count of distinct elements would be useful.

Example 3.8.4. *U is the set of all possible phone numbers, and $\{z_j\}_{j=1}^N$ is a list of phone numbers which have communicated with a particular cellphone tower over some period of time. The cardinality of the sequence as a set would be the number of unique cellphones that used the tower.*

Example 3.8.5. *U is all possible pairs of words in the English language. The sequence $\{z_j\}_{j=1}^N$ is a list of all pairs of words that appear in a user's current email outbox. The cardinality of the sequence as a set would be an indicator of the variation of a given user's word choice in writing emails.*

We consider two naive solutions to this problem, and observe how they do not achieve the stated goal.

Algorithm 19 Naive Distinct Elements by D-array

Input: $\{z_j\}_{j=1}^N$
Output: number of distinct elements in $\{z_j\}_{j=1}^N$
Let $A :=$ array of zeros of size D
for $j = 1$ to N **do**
  if $A[z_j] = 0$ **then**
    $A[z_j] := 1$
  end if
end for
$\|A\|_0$

Algorithm 20 Naive Distinct Elements by Sorted List

Input: $\{z_j\}_{j=1}^N$
Output: number of distinct elements in $\{z_j\}_{j=1}^N$
Let $L[j] := z_j$
Sort L
for $j = 1$ to $N - 1$ **do**
  if $L[j] = L[j + 1]$ **then**
    flag $L[j + 1]$ for removal
  end if
end for
$|L|$

However, neither of these algorithms meet the requirements of the stated goal. In the case of algorithm 19 the array A clearly occupies D bits. In the case of 20, the list L needs N entries, and so will occupy at least N bits of memory. In a future lecture, we will study

the Flajolet-Martin Algorithm which does solve the distinct element problem with constant memory.

3.8.3 Distinct elements: Flajolet Martin algorithm

Recall that we have a sequence $\{z_i\}_{i=0}^{N-1}$ and $\forall j, z_j \in U$, a subset of $\mathbb{Z}$ where $|U| = M$.
Problem: How many distinct elements are there in the sequence ?

Goal. *Solve this problem using* $o(\min(M, N))-$*memory.*

Solution. *Flajolet-Martin Algorithm.*

Algorithm 21 Flajolet-Martin Algorithm

Input: $z_1, \ldots, z_N \in [M]$, KL i.i.d. perfect hash functions $h_{(k,\ell)} : [M] \to [0, 1]$
Output: $\tilde{E}$ estimate of $|\{z_1, \ldots, z_N\}| = \tilde{n}$

$E^{(k,\ell)} \leftarrow 1$, $\forall k \in [K], \ell \in [L]$
for $j = 1, \ldots, N$ **do**
  for $\ell = 1, \ldots, L$ **do**
    for $k = 1, \ldots, K$ **do**
      $E^{(k,\ell)} \leftarrow \min\left(E^{(k,\ell)}, h_{(k,\ell)}(z_j)\right)$
    end for
    $E^{\ell} \leftarrow \frac{1}{K} \sum_{k=1}^{K} E^{(k,\ell)}$
  end for
end for
$E \leftarrow \text{median}\,(E_1, \ldots, E_L)$
$\tilde{E} \leftarrow \frac{1}{E} - 1$

- $\forall j, l$ we have:

  ???SWAP k's and j's???

$$\begin{aligned} E^{(j,\ell)} &= \min\left\{h_{(j,\ell)}(z_k)\right\}_{k\in[N]} \\ &= \min\left\{h_{(j,\ell)}(a) | a \in \{z_0, z_1, \ldots, z_{N-1}\}\right\} \\ &= \min \tilde{n}, \qquad \tilde{n} \text{ are i.i.d uniform random variable from } [0, 1]. \end{aligned}$$

- The random variables $\tilde{n}$ are order statistics.

$\rightsquigarrow$ Essentially the $F - M$ algorithm is transforming a distinct count problem into the estimation of an order statistics.

Lemma 3.8.6. *The probability density of* $E^{(k,\ell)} = \min\{u_1, \ldots, u_{\tilde{n}}\}$ *where* $u_1, \ldots, u_{\tilde{n}}$ *are i.i.d uniform in the interval* $[0, 1]$ *is* $p(x) = \tilde{n}(1 - x)^{\tilde{n}-1}$

Proof.

$$\begin{aligned}
\int_0^y p(x)dx &= \mathbb{P}\left[\min\left\{u_1,\ldots,u_{\tilde{n}}\right\} \in [0,y]\right] \\
&= 1 - \mathbb{P}\left[\min\left\{u_1,\ldots,u_{\tilde{n}}\right\} \in [y,1]\right] \\
&= 1 - \mathbb{P}\left[\text{all } u_j \in [y,1]\right] \\
&= 1 - \prod_{\ell=1}^{\tilde{n}} \mathbb{P}\left[u_\ell \in [y-1]\right] \qquad \text{using independence} \\
&= 1 - \big(1-y\big)^{\tilde{n}} \qquad \text{since } u_\ell \sim \mathcal{U}[0,1].
\end{aligned}$$

By the Fundamental Theorem of Calculus, we have that:

$$p(x) = \tilde{n}\big(1-x\big)^{\tilde{n}-1}.$$

□

Lemma 3.8.7. *If* $E^{(k,\ell)} = \min\left\{u_1,\ldots,u_{\tilde{n}}\right\}$ *then:*

$$\mathbb{E}\left[E^{(k,\ell)}\right] = (\tilde{n}+1)^{-1} \qquad \text{Var}\left[E^{(k,\ell)}\right] = \frac{\tilde{n}}{(\tilde{n}+1)^2(\tilde{n}+2)} < \frac{1}{\tilde{n}+1}^2.$$

Proof.

$$\begin{aligned}
\mathbb{E}\left[E^{(k,\ell)}\right] &= \int_0^1 xp(x)dx \\
&= \int_0^1 x\tilde{n}\big(1-x\big)^{\tilde{n}-1}dx \\
&= \int_0^1 \tilde{n}\left[y^{\tilde{n}-1} - y^{\tilde{n}}\right]dy \qquad \text{using } y = 1-x \\
&= \tilde{n}\left[\frac{y^{\tilde{n}}}{\tilde{n}} - \frac{y^{\tilde{n}+1}}{\tilde{n}+1}\right]_0^1 \\
&= \frac{1}{\tilde{n}+1}.
\end{aligned}$$

Also

$$\begin{aligned}
\mathbb{E}\left[\left(E^{(k,\ell)}\right)^2\right] &= \int_0^1 x^2 p(x) dx \\
&= \int_0^1 x^2 \tilde{n}\big(1-x\big)^{\tilde{n}-1} dx \\
&= \int_0^1 \tilde{n}\left[y^{\tilde{n}-1} - 2y^{\tilde{n}} + y^{\tilde{n}+1}\right] dy \qquad \text{using } y = 1-x \\
&= \tilde{n}\left[\frac{y^{\tilde{n}}}{\tilde{n}} - 2\frac{y^{\tilde{n}+1}}{\tilde{n}+1} + \frac{y^{\tilde{n}+2}}{\tilde{n}+2}\right]_0^1 \\
&= \frac{1}{(\tilde{n}+1)(\tilde{n}+2)}.
\end{aligned}$$

Thus:

$$\begin{aligned}
\mathrm{Var}\left[E^{(k,\ell)}\right] &= \mathbb{E}\left[\left(E^{(k,\ell)}\right)^2\right] - \left(\mathbb{E}\left[E^{(k,\ell)}\right]\right)^2 \\
&= \frac{\tilde{n}}{(\tilde{n}+1)^2(\tilde{n}+2)} < \frac{1}{(\tilde{n}+1)^2}.
\end{aligned}$$

□

Since we have the expectation and variance of a single estimator $E_{(k,\ell)}$ we can use reasoning similar to that seen in Monte Carlo integration and Morris' Algorithm to combine estimates in a median of means scheme to achieve the following overall estimate with

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} E^{(k,\ell)}\right] = \frac{1}{\tilde{n}+1}, \mathrm{Var}\left[\frac{1}{K}\sum_{k=1}^{K} E^{(k,\ell)}\right] \leq \frac{1}{K(\tilde{n}+1)^2}$$

By choosing $K = \frac{10}{\epsilon^2}$ and using Chebyshev inequality then we have that

$$P\left[\left|\frac{1}{K}\sum_{k=1}^{K} E^{(k,\ell)} - \frac{1}{\tilde{n}+1}\right| < \epsilon\left(\frac{1}{\tilde{n}+1}\right)\right] > 0.9$$

Now, choosing $L \geq c\log\left(\frac{1}{q}\right)$ and using the sum of indicator variables and Chernoff inequality, we can argue that the majority of the estimators will be within our chosen error bound with high probability

$$P\left[\left|\tilde{E} - \frac{1}{\tilde{n}+1}\right| < \epsilon\left(\frac{1}{\tilde{n}+1}\right)\right] \geq 1-q$$

for $q \in (0,1)$. More formally we have the following theorem

Theorem 3.8.8 (Flajolet-Martin Algorithm)**.** *Choose $\epsilon, q \in (0,1)$. Then Algorithm 21 will output an estimate $\tilde{E} = \frac{1}{E} - 1$ satisfying*

$$\frac{\tilde{n}}{1+\epsilon} - \left(\frac{\epsilon}{1+\epsilon} - 1\right) \leq \tilde{E} \leq \frac{\tilde{n}}{1-\epsilon} + \left(\frac{\epsilon}{1-\epsilon} - 1\right)$$

with probability at least $1 - q$.

Proof. The formal proof is left as an exercise to the reader. □

Exercise 3.8.1. *Prove Lemma 3.8.7*

Exercise 3.8.2. *Using the discussion in this section, formalize a proof of Theorem 3.8.8*

3.9 Notes and References

non-measure theoretic here – other options?

3.10 TO INCLUDE – USE TO PROVE EXISTENCE OF JL MAPS AS FAST AS POSSIBLE

3.11 Useful General Purpose Probability Inequalities (MTH 994 Lecture 5)

Theorem 3.11.1 (Cramer's Theorem)**.** *Let $X_1, \ldots, X_m$ be a sequence of independent real-valued random variables with cumulant generating functions*

$$C_{X_\ell}(\theta) = \ln\left(\mathbb{E}\left[e^{\theta X_\ell}\right]\right)$$

where $\ell \in [m]$. Then, $\forall t > 0$

$$P\left[\sum_{\ell=1}^{m} X_\ell \geq t\right] \leq \exp\left(\inf_{\theta > 0}\left\{-\theta t + \sum_{\ell=1}^{m} C_{X_\ell}(\theta)\right\}\right)$$

Proof. By Markov's inequality, for any $\theta > 0$, and independence of the random variables

we have:

$$\begin{aligned} P\left[\sum_{\ell=1}^{m} X_\ell \geq t\right] &= P\left[\exp\left(\theta \sum_{\ell=1}^{m} X_\ell\right) \geq \exp(\theta t)\right] \\ &\leq e^{-\theta t}\mathbb{E}\left[\exp\left(\theta \sum_{\ell=1}^{m} X_\ell\right)\right] \\ &= e^{-\theta t}\prod_{\ell=1}^{m}\mathbb{E}\left[\exp(\theta X_\ell)\right] \\ &= \exp\left(\ln\left(e^{-\theta t}\prod_{\ell=1}^{m}\mathbb{E}\left[\exp(\theta X_\ell)\right]\right)\right) \\ &= \exp\left(-\theta t + \sum_{\ell=1}^{m} C_{X_\ell}(\theta)\right) \end{aligned}$$

Since the above holds for all $\theta > 0$, it will hold for the infimum, which matches our desired outcome □

Theorem 3.11.2 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_m$ be a sequence of independent random variables such that $\mathbb{E}[X_\ell] = 0$ and $|X_\ell| \leq B_\ell$, $\forall \ell \in [m]$. Then, $\forall t > 0$*

$$P\left[\sum_{\ell=1}^{m} X_\ell \geq t\right] \leq \exp\left(\frac{-t^2}{2\sum_{\ell=1}^{m} B_\ell^2}\right)$$

and so

$$P\left[\left|\sum_{\ell=1}^{m} X_\ell\right| \geq t\right] \leq 2\exp\left(\frac{-t^2}{2\sum_{\ell=1}^{m} B_\ell^2}\right)$$

Proof. We first estimate the moment generating function $\mathbb{E}[\exp(\theta X_\ell)]$ and then apply Cramer's Theorem.

For some $\tilde{t}_\ell > 0$, we can write each of the random variables as some combination of its bounds:

$$X_\ell = \tilde{t}(-B_\ell) + (1 - \tilde{t})B_\ell$$

Solving for $\tilde{t}_\ell$ we have

$$\tilde{t}_\ell = \frac{B_\ell - X_\ell}{2B_\ell} \in [0, 1]$$

Since $\exp(\theta x)$, $\theta > 0$ is a convex function and so we have the bound

$$\begin{aligned}\exp(\theta X_\ell) &\leq \tilde{t}\exp(-\theta B_\ell) + (1-\tilde{t})\exp(\theta B_\ell) \\ &= \left(\frac{B_\ell - X_\ell}{2B_\ell}\right)\exp(-\theta B_\ell) + \left(\frac{B_\ell + X_\ell}{2B_\ell}\right)\exp(\theta B_\ell)\end{aligned}$$

and so taking the expectation and recalling $\mathbb{E}[X_\ell] = 0$, we obtain the moment generating function and the following bound

$$\begin{aligned}\mathbb{E}\left[\exp(\theta X_\ell)\right] &\leq \frac{1}{2}\left[\exp(-\theta B_\ell) + \exp(\theta B_\ell)\right] \\ &= \sum_{k=0}^{\infty}\frac{(\theta B_\ell)^{2k}}{(2k)!} \\ &\leq \sum_{k=0}^{\infty}\frac{(\theta B_\ell)^{2k}}{2^k k!} \\ &= \exp\left(\frac{\theta^2 B_\ell^2}{2}\right)\end{aligned}$$

Apply Cramer's Theorem 3.11.1 with $\theta = \frac{t}{\sum_{\ell=1}^m B_\ell^2}$ and bound C_{X_ℓ} by $\frac{\theta^2 B_\ell^2}{2}$ to obtain

$$P\left[\sum_{\ell=1}^m X_\ell \geq t\right] \leq \exp\left(\frac{-t^2}{2\sum_{\ell=1}^m B_\ell^2}\right)$$

□

Definition 3.11.3 (Radamacher Random Variable)**.** *A random variable X such that*

$$X = \begin{cases} 1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases}$$

Note that the expectation of such a variable is 0.

Corollary 3.11.4. *Let $a \in \mathbb{R}^m$ and $\mathbf{X} = (X_1, \ldots, X_m)$ be a random vector with i.i.d. Radamacher entries. Then, $\forall u > 0$, we have*

$$P\left[\left|\sum_{\ell=1}^m a_\ell X_\ell\right| \geq u\|\mathbf{a}\|_2\right] \leq 2\exp\left(-\frac{u^2}{2}\right)$$

The corollary follows from an application of Hoeffding's inequality.

Theorem 3.11.5 (Bernstein's Inequality)**.** *Let $X_1, \ldots, X_m$ be a sequence of independent random variables such that $\mathbb{E}[X_\ell] = 0$ such that $\forall n \geq 2$*

$$\mathbb{E}[|X_\ell|^n] \leq n! R^{n-2} \sigma_\ell^2 / 2, \, \forall \ell \in [m]$$

for some constants $R > 0$ and $\sigma_\ell > 0, \ell \in [m]$. Then $\forall t > 0$

$$P\left[\sum_{\ell=1}^{m} X_\ell \geq t\right] \leq 2 \exp\left(\frac{-t^2/2}{\sigma^2 + Rt}\right)$$

where $\sigma^2 = \sum_{\ell=1}^m \sigma_\ell^2$

Proof. First we bound the moment generating function $\mathbb{E}[\exp(\theta X_\ell)]$ by expanding the exponential function into a series and applying the linearity of expectation after exchanging integration and summation (Fubini and Dominated Convergence):

$$\begin{aligned}
\mathbb{E}[\exp(\theta X_\ell)] &= 1 + \theta \mathbb{E}[X_\ell] + \sum_{n=2}^{\infty} \frac{\theta^n \mathbb{E}[X_\ell^n]}{n!} \\
&= 1 + \sum_{n=2}^{\infty} \frac{\theta^n \mathbb{E}[X_\ell^n]}{n!} \\
&\leq 1 + \frac{\sigma_\ell^2 \theta^2}{2} \sum_{n=0}^{\infty} (\theta R)^n \\
&\leq 1 + \frac{\sigma_\ell^2 \theta^2}{2} (1 - R\theta)^{-1} \\
&\leq \exp\left(\frac{\sigma_\ell^2 \theta^2}{2(1 - R\theta)}\right)
\end{aligned}$$

where we require that $0 < R\theta < 1$ to get convergence of the geometric series. Apply Cramer's theorem then using this bound on the cumulant density functions to obtain

$$\begin{aligned}
P\left[\left|\sum_{\ell=1}^{m} X_\ell\right| \geq t\right] &\leq 2 \inf_{\theta \in (0, R^{-1})} \exp\left(-\theta t + \sum_{\ell=1}^{m} \frac{\sigma_\ell^2 \theta^2}{2(1 - R\theta)}\right) \\
&= 2 \inf_{\theta \in (0, R^{-1})} \exp\left(-\theta t + \frac{\sigma^2 \theta^2}{2(1 - R\theta)}\right)
\end{aligned}$$

Choosing $\theta = \frac{t}{\sigma^2 + Rt} < \frac{1}{R}$ then leads to our desired final bound. □

Lemma 3.11.6. $\mathbb{E}[|X|^n] \leq n \int_0^\infty P[|X| \geq t] t^{n-1} dt, \, \forall n > 0$

Proof.

$$\begin{aligned}
\int_\Omega |X|^n p(x)dx &= \int_\Omega \left(\int_0^\infty \mathbb{1}_{\{0\le y\le |X|^n\}} dy \right) p(x)dx \\
&= \int_0^\infty \int_\Omega P\left[|X|^n \ge y\right] dy && \text{Fubini's Theorem} \\
&= n\int_0^\infty \int_\Omega P\left[|X|^n \ge t^n\right] t^{n-1} dt && \text{change of variable } y = t^n \\
&= n\int_0^\infty \int_\Omega P\left[|X| \ge t\right] t^{n-1} dt
\end{aligned}$$

□

We now introduce some definitions about a large class of random variables which will help us prove in a general way many important and useful results for JL maps.

Definition 3.11.7 (Sub-exponential Random Variable)**.** *We say that $X \in \mathbb{R}$ is sub-exponential random variable if $\exists \beta, \kappa > 0$ such that*

$$P\left[|X| \ge t\right] \le \beta e^{-\kappa t}, \forall t > 0$$

We can understand this as saying that the random variable decays exponentially.

Definition 3.11.8 (Sub-gaussian Random Variable)**.** *We say that $X \in \mathbb{R}$ is subgaussian random variable if $\exists \beta, \kappa > 0$ such that*

$$P\left[|X| \ge t\right] \le \beta e^{-\kappa t^2}, \forall t > 0$$

Again, we understand this as saying that the random variable decays faster than exponential, at a rate comparable to a Gaussian. What types of random variables fit these definitions? The following examples provide some indication as to the richness of the classification.

Example 3.11.9. *If X is sub-gaussian, then X^2 is sub-exponential with the same β and κ:*

$$\beta e^{-\kappa t^2} \ge P\left[|X| \ge t\right] = P\left[|X|^2 \ge t^2\right]$$

but after a relabeling of t we rewrite as

$$\beta e^{-\kappa t} \ge P\left[X^2 \ge t\right]$$

for all $t > 0$.

Example 3.11.10. *All bounded random variables, e.g. Radamacher, Bernoulli, uniform on bounded interval, all discrete random variables, are subgaussian.*

Example 3.11.11. *A Gaussian random variable $X \sim \mathcal{N}(0,1)$ is sub-gaussian with $\beta = 1$ and $\kappa = 1/2$*

Theorem 3.11.12 (Bernstein's Inequality for sub-expoential random variables)**.** *Let $X_1, \ldots, X_m$ be independent mean 0 sub-expoential random variables such that $\exists \beta, \kappa > 0$ where $P[|X_\ell| \geq t] \leq \beta e^{-\kappa t}$, $\forall t > 0, \forall \ell \in [m]$. Then*

$$P\left[\left|\sum_{\ell=1}^{m} X_\ell\right| \geq t\right] \leq 2\exp\left(\frac{-(\kappa t)^2/2}{2\beta m + \kappa t}\right)$$

Proof. By Lemma 3.11.6 we have

$$\begin{aligned}
\mathbb{E}\left[|X_\ell|^n\right] &= n\int_0^\infty P\left[|X| \geq t\right] t^{n-1} dt \\
&= \beta n \int_0^\infty e^{-\kappa t} t^{n-1} dt && \text{let } \kappa t = u \\
&= \beta n \kappa^{-n} \int_0^\infty e^{-u} u^{n-1} du && \text{Notice } \Gamma \text{ function definition} \\
&= \beta \kappa^{-n} n!
\end{aligned}$$

Now apply Bernstein's Inequality 3.11.5 with $R = \kappa^{-1}$ and $\sigma_\ell^2 = 2\beta\kappa^{-2}$ to obtain the desired bound. □

Exercise 3.11.1. *Prove that if X is a bounded random variable then $\exists t_0 \in \mathbb{R}^+$ such that X is sub-gaussian $\forall \beta, \kappa > 0$ satisfying $t_0 = \sqrt{\frac{\ln \beta}{\kappa}}$*

Exercise 3.11.2. *Prove that if $X \sim \mathcal{N}(0,1)$ then*

$$P\left[|X| \geq t\right] \leq e^{-t^2/2}, \forall t > 0$$

Exercise 3.11.3. *Let X be uniformly distributed on $[-1,1]$ show that*

$$\mathbb{E}\left[|X|^2\right] = 1/3$$

and that

$$\mathbb{E}\left[\exp\left(\theta X\right)\right] \leq \exp\left(\theta^2/6\right) = \exp\left(\theta^2 \mathbb{E}[|X|^2]/2\right)$$

3.12 Stability of Subgaussians as a Class of Random Variables

Lemma 3.12.1. *If X is a sub-gaussian random variable with parameters $\beta, \kappa > 0$ then*

$$\|X\|_p = \left(\mathbb{E}\left[|X|^p\right]\right)^{1/p} \leq \kappa^{-1}\beta^{1/2} p^{1/2}, \forall p \geq 1$$

Proof. By Lemma 3.11.6

$$\begin{aligned}
\mathbb{E}\left[|X_\ell|^p\right] &= p\int_0^\infty P\left[|X| \geq t\right] t^{p-1} dt \\
&= \frac{p}{(2\kappa)^{p/2}} \int_0^\infty P\left[|X| \geq \frac{u}{\sqrt{2\kappa}}\right] u^{p-1} du && \text{let } t = \frac{u}{\sqrt{2\kappa}} \\
&\leq \frac{p\beta}{(2\kappa)^{p/2}} \int_0^\infty e^{-u^2/2} u^{p-1} du && X \text{is sub-gaussian} \\
&= \frac{p\beta}{2\kappa^{p/2}} \Gamma\left(\frac{p}{2}\right) \\
&= \frac{p\beta}{\kappa^{p/2}} \sqrt{\frac{\pi}{2}} \left(\frac{p}{2}\right)^{p/2-1/2} e^{-p/2} e^{1/6p} \Gamma\left(\frac{p}{2}\right) \\
&= \kappa^{-p/2} \beta p^{p/2} \left[\frac{1}{2^{p/2}} \sqrt{p\pi} e^{-p/2+1/6p}\right]
\end{aligned}$$

$\left[\frac{1}{2^{p/2}} \sqrt{p\pi} e^{-p/2+1/6p}\right]$ is monotonically decreasing for $p \geq 1$ and is bounded by 1 from above, which then taking p-th root leads to the desired final bound. □

Lemma 3.12.2. *If X is sub-gaussian with parameters β, κ then $\exists c \in (0, \kappa)$ and $\tilde{c} \geq 1 + \frac{\beta c \kappa^{-1}}{1 - c\kappa^{-1}}$ such that $\mathbb{E}\left[\exp\left(cX^2\right)\right] \leq \tilde{c}$.*

Proof. By using Lemma 3.11.6 with $p \leftarrow 2n$ then

$$\mathbb{E}\left[|X|^{2n}\right] \leq \beta \kappa^{-n} n!$$

$$\begin{aligned}
\mathbb{E}\left[\exp\left(cX^2\right)\right] &= \int_0^\infty \sum_{n=0}^\infty \frac{c^n X^{2n}}{n!} p(x) dx \\
&= \sum_{n=0}^\infty \frac{c^n \mathbb{E}\left[X^{2n}\right]}{n!} && \text{Fubini's Theorem} \\
&\leq 1 + \sum_{n=1}^\infty c^n \beta \kappa^{-n} && \text{Series converges when } c \in (0, \kappa) \\
&= 1 + \frac{\beta c \kappa^{-1}}{1 - c\kappa^{-1}}
\end{aligned}$$

□

Theorem 3.12.3 (Alternative Characterization of Sub-gaussian Property)**.** *Let $X \in \mathbb{R}$ be a random variable*

1. *If X is sub-gaussian with $\mathbb{E}[X] = 0$ then $\forall c \in \mathbb{R}^+$ with $c > \max\left\{\frac{1}{2\kappa} + \frac{4e^2}{\kappa}\ln(1+\beta), \frac{\sqrt{2}\beta e^2}{\kappa\sqrt{\pi}}\right\}$ then*

$$\mathbb{E}\left[\exp(\theta X)\right] \leq \exp\left(c\theta^2\right), \forall \theta \in \mathbb{R}^+ \tag{3.15}$$

2. *If 3.15 holds for some $c \in \mathbb{R}^+$ then $\mathbb{E}[X] = 0$ and X is sub-gaussian with parameters $\beta = 2$ and $\kappa = \frac{1}{4c}$*

Proof. We begin with the second part of the statement of the theorem. Assume inequality 3.15 holds.

$$\begin{aligned} P[X \geq t] &= P\left[\exp\left(\theta X \geq e^{\theta t}\right)\right] \\ &\leq e^{-\theta t}\mathbb{E}\left[\exp(\theta X)\right] && \text{Markov's Inequality} \\ &\leq e^{c\theta^2 - \theta t} && \text{by hypothesis} \end{aligned}$$

Setting $\theta = \frac{t}{2c}$ minimizes the right hand side and we have $P[X \geq t] \leq e^{\frac{-t^2}{4c}}$. We can repeat the same argument to conclude that $P[-X \geq t] \leq e^{\frac{-t^2}{4c}}$ and so conclude by a union bound that $P[|X| \geq t] \leq 2e^{\frac{-t^2}{4c}}$, i.e. X is sub-gaussian with with parameters $\beta = 2$ and $\kappa = \frac{1}{4c}$.

To see that the random variable must have mean zero, recall the bounds $(1+x) \leq e^x, \forall x \in \mathbb{R}$. Use this bound, taking expectation with respect to the random variable X, and using the series definition of the exponential we have

$$\begin{aligned} 1 + \mathbb{E}[\theta X] &\leq \mathbb{E}\left[\exp(\theta X)\right] \\ \implies 1 + \theta\mathbb{E}[X] &\leq \exp\left(c\theta^2\right) \\ \implies \theta\mathbb{E}[X] &\leq \frac{1}{2}c\theta^2 + \mathcal{O}\left(\theta^4\right) \end{aligned}$$

So sending $\theta \to 0$ yields $\mathbb{E}[X] = 0$.

Now we consider the first part of the theorem. Assume that the random variable X is sub-gaussian with $c > \max\left\{\frac{1}{2\kappa} + \frac{4e^2}{\kappa}\ln(1+\beta), \frac{\sqrt{2}\beta e^2}{\kappa\sqrt{\pi}}\right\}$. For the moment, consider $|\theta| \leq \theta_0$ for some yet to be determined θ_0

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\theta X\right)\right] &= 1 + \theta\mathbb{E}\left[X\right] + \sum_{n=2}^{\infty} \frac{\theta^n \mathbb{E}\left[X^n\right]}{n!} && \text{Fubini, linearity of expectation} \\
&= 1 + \sum_{n=2}^{\infty} \frac{\theta^n \mathbb{E}\left[X^n\right]}{n!} && \text{mean zero} \\
&\leq 1 + \sum_{n=2}^{\infty} \frac{|\theta|^n \kappa^{n/2} \beta n^{n/2}}{\sqrt{2\pi} n^n e^{-n}} && \text{Lemma 3.12.1. Sterling's formula} \\
&= 1 + \frac{\beta}{\sqrt{2\pi}} \frac{\theta^2 e^2}{\kappa} \sum_{n=0}^{\infty} \theta_0^n \kappa^{-n/2} e^n && \text{re-indexing, } |\theta| \leq \theta_0 \\
&= 1 + \theta^2 \frac{\beta}{\sqrt{2\pi}} \frac{\theta^2 e^2}{\kappa} \frac{1}{1 - \frac{1}{2}} && \text{set } \theta_0 = \frac{\sqrt{\kappa}}{2e} \\
&\leq \exp(c\theta^2) && \text{when } c > \frac{\sqrt{2}\beta e^2}{\kappa\sqrt{\pi}}
\end{aligned}$$

We now must consider the case when $|\theta| > \theta_0$. We wish to show that

$$\mathbb{E}\left[\exp\left(\theta X\right)\right] \leq \exp\left(c\theta^2\right) \iff \mathbb{E}\left[\exp\left(\theta X - c\theta^2\right)\right] \leq 1$$

Notice that by completing the square, for any positive constant c

$$\theta X - c\theta^2 = -\left(\sqrt{c}\theta - \frac{X}{2\sqrt{c}} + \frac{X^2}{4c}\right) \leq \frac{X^2}{4c}$$

So then

$$\mathbb{E}\left[\exp\left(\theta X - c\theta^2\right)\right] \leq \mathbb{E}\left[\exp\left(\frac{X^2}{4c}\right)\right]$$

In particular for constant $\frac{1}{2\kappa}$ then,

$$\mathbb{E}\left[\exp\left(\theta X - \frac{1}{2\kappa}\theta^2\right)\right] \leq \mathbb{E}\left[\exp\left(\frac{\kappa X^2}{2}\right)\right]$$

So then in Lemma 3.12.2, where for $c = \frac{\kappa}{2}$ we have for $\tilde{c} > 1 + \frac{\beta c \kappa^{-1}}{1 - c\kappa^{-1}} = 1 + \beta$. Noting this then we have that

$$\mathbb{E}\left[\exp\left(\frac{\kappa}{2}X^2\right)\right] \leq 1 + \beta$$

So combining the two inequalities we have

$$\mathbb{E}\left[\exp\left(\theta X\right)\right] \leq \exp\left(\frac{\theta^2}{2\kappa}\right)(1 + \beta)$$

Now let $\rho = \ln(1+\beta)\,\theta_0^{-2}$

$$\begin{aligned}
\mathbb{E}\left[\exp(\theta X)\right] &\leq (1+\beta)\exp\left(\frac{\theta^2}{2\kappa}\right) \\
&= (1+\beta)\exp\left(-\rho\theta^2\right)\exp\left(\frac{\theta^2}{2\kappa}\right)\exp\left(\rho\theta^2\right) \\
&\leq (1+\beta)\exp\left(-\rho\theta_0^2\right)\exp\left(\left(\frac{1}{2\kappa}+\rho\right)\theta^2\right) \\
&\leq \exp\left(\left(\frac{1}{2\kappa}+\rho\right)\theta^2\right)
\end{aligned}$$

Noting that $\theta_0 = \frac{\sqrt{\kappa}}{2e}$ and $\rho = \frac{4e^2}{\kappa}\ln(1+\beta)$ we have then the desired bound. □

Theorem 3.12.4 (Stability of Sub-gaussians)**.** *Let $\mathbf{X} = X_1, \ldots, X_m$ be independent mean zero sub-gaussian random variables such that $\mathbb{E}\left[\exp(\theta X_\ell)\right] \leq \exp\left(c\theta^2\right)$ for $\ell \in [m], \theta \in \mathbb{R}^+$. Let $\mathbf{a} \in \mathbb{R}^m$ and define $z = \langle \mathbf{a}, X\rangle$. Then z is sub-gaussian with*

1. $$\mathbb{E}\left[\exp(\theta z)\right] \leq \exp\left(c\|\mathbf{a}\|_2^2\theta^2\right)$$

2. $$P\left[|z| \geq t\right] \leq 2\exp\left(\frac{-t^2}{4c\|\mathbf{a}\|_2^2}\right), \forall t > 0$$

Proof. 1.

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\theta\sum_{\ell=1}^m a_\ell X_\ell\right)\right] &= \prod_{\ell=1}^m \left[\exp(\theta a_\ell X_\ell)\right] && X_\ell \text{ independent} \\
&\leq \prod_{\ell=1}^m \exp\left(ca_\ell^2\theta^2\right) \\
&\leq \exp\left(c\|\mathbf{a}\|_2^2\theta^2\right)
\end{aligned}$$

2. This follows from part 2 of Theorem 3.12.3.

□

Definition 3.12.5. *A sub-gaussian random variable X allows a parameter c if*

$$\mathbb{E}\left[\exp(\theta X)\right] \leq \exp\left(c\theta^2\right), \forall\theta \in \mathbb{R}^+$$

Lemma 3.12.6. *Let $\mathbf{Z} \in \mathbb{R}^N$ be a random vector with independent, mean zero, variance 1, sub-gaussian entries that all allow the same parameter $c \in \mathbb{R}^+$. Then*

1. $$\mathbb{E}\left[|\langle \mathbf{Z}, \mathbf{x}\rangle|^2\right] = \|\mathbf{x}\|_2^2, \ \forall \mathbf{x} \in \mathbb{R}^N$$

2. *$\langle \mathbf{Z}, \mathbf{x}/\|\mathbf{x}\|_2\rangle$ is sub-gaussian and also allows the parameter c*

Proof. 1. We expand the square of the sum, use linearity of expectation, independence of variables and the mean zero and variance of one of all the random variables to obtain:

$$\begin{aligned}
\mathbb{E}\left[|\langle \mathbf{Z}, \mathbf{x}\rangle|^2\right] &= \mathbb{E}\left[\left(\sum_{\ell=1}^N Z_\ell x_\ell\right)^2\right] \\
&= \mathbb{E}\left[\sum_{\ell=1}^N \sum_{k=1}^N Z_\ell Z_k x_\ell x_k\right] \\
&= \sum_{\ell=1}^N \sum_{k=1}^N \mathbb{E}\left[Z_\ell Z_k\right] x_\ell x_k \\
&= \sum_{\ell=1}^N \mathbb{E}\left[Z_\ell^2\right] x_\ell^2 + \sum_{\ell=1}^N \sum_{k\neq \ell} \mathbb{E}\left[Z_\ell\right] \mathbb{E}\left[Z_k\right] x_\ell x_k \\
&= \sum_{\ell=1}^N (1) x_\ell^2 \\
&= \|\mathbf{x}\|_2^2
\end{aligned}$$

2. Follows from part 1. of Theorem 3.12.4

□

Theorem 3.12.7 (Concentration Inequality for Sub-gaussian Random Variables)**.** *Let $\Phi \in \mathbb{R}^{m\times N}$ be a matrix with independent, mean zero, variance one sub-gaussian entries that all allow parameter c. Then $\forall \mathbf{x} \in \mathbb{R}^n$ and $t \in (0,1)$,*

$$P\left(\left|m^{-1}\|\Phi\mathbf{x}\|_2^2 - \|\mathbf{x}\|_2^2\right| \geq t\|\mathbf{x}\|_2^2\right) \leq 2\exp\left(-\tilde{c}mt^2\right)$$

where $\tilde{c}$ depends only on c, $\tilde{c} = \frac{1}{8c(16c+1)}$

Proof. Let $\mathbf{Y}_1, \ldots, \mathbf{Y}_m \in \mathbb{R}^N$ be the rows of the matrix Φ. Define

$$Z_\ell = |\langle \mathbf{Y}_\ell, \mathbf{x}\rangle|^2 - \|\mathbf{x}\|_2^2, \ \ell \in [m].$$

By Lemma 3.12.6 we have that $\mathbb{E}[Z_\ell] = 0$. Furthermore, $\langle \mathbf{Y}_\ell, \mathbf{x}/\|\mathbf{x}\|_2 \rangle$ is sub-gaussian with parameter c. Now using the characterization of sub-gaussian random variables seen in Theorem 3.12.3, we have that $\langle \mathbf{Y}_\ell, \mathbf{x}/\|\mathbf{x}\|_2 \rangle$ works as a sub-gaussian random variable for $\beta = 2$ and $\kappa = 1/4c$ with mean 0.

Therefore

$$P\left[|\langle \mathbf{Y}_\ell, \mathbf{x}/\|\mathbf{x}\|_2 \rangle| \geq r\right] \leq \beta e^{-\kappa r^2}$$

Squaring the random variable then gives us a sub-exponential random variable concentration result,

$$P\left[|\langle \mathbf{Y}_\ell, \mathbf{x}/\|\mathbf{x}\|_2 \rangle|^2 \geq \tilde{r}\right] \leq \beta e^{-\kappa \tilde{r}}$$

where $\tilde{r} = r^2$. Note

$$\begin{aligned}
\frac{1}{\|\mathbf{x}\|_2}\left(m^{-1}\|\Phi\mathbf{x}\|_2^2 - \|\mathbf{x}\|_2^2\right) &= \frac{1}{m}\sum_{\ell=1}^{m}\left(\frac{|\langle \mathbf{Y}_\ell, \mathbf{x}\rangle|^2 - \|\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}\right) \\
&= \frac{1}{m\|\mathbf{x}\|_2^2}\sum_{\ell=1}^{m} Z_\ell
\end{aligned}$$

We now have what we need to satisfy Bernstein's inequality for sub-exponential random variables. That is

$$\begin{aligned}
P\left[\frac{1}{m\|\mathbf{x}\|_2^2}\left|\sum_{\ell=1}^{m} Z_\ell\right| \geq t\right] &= P\left[\left|\sum_{\ell=1}^{m}\frac{Z_\ell}{\|\mathbf{x}\|_2^2}\right| \geq mt\right] \\
&\leq 2\exp\left(\frac{-mt^2\kappa^2}{4\beta + 2\kappa t}\right) \\
&\leq 2\exp\left(-mt^2\tilde{c}\right)
\end{aligned}$$

when $\beta = 2, \kappa = \frac{1}{4c}$ and $\tilde{c} = \frac{1}{8c(16c+1)}$ □

Theorem 3.12.8. *Let $S \subset \mathbb{R}^N$ be an arbitrary finite subset of $\mathbb{R}^N$. Let $p, \epsilon \in (0,1)$. Finally, let $\Phi \in \mathbb{R}^{m\times N}$ be a matrix with independent, mean zero, variance one, sub-gaussian entries all allowing the parameter c. Then*

$$(1-\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \|\frac{1}{\sqrt{m}}\Phi(\mathbf{x}-\mathbf{y})\|_2^2 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2$$

will hold for all $\mathbf{x}, \mathbf{y} \in S$ with probability at least p, provided that $m \geq \frac{8c(16c+1)}{\epsilon^2}\ln\left(\frac{|S|^2}{1-p}\right)$.

Proof. The proof of this theorem is left as an exercise to the reader. □

Theorem 3.12.9. *Let $S \subset \mathbb{C}^N$ be an arbitrary finite subset of $\mathbb{C}^N$, $p, \epsilon \in (0,1)$. Then Φ as in Theorem 3.12.8 will satisfy*

$$(1-\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \|\frac{1}{\sqrt{m}}\Phi(\mathbf{x}-\mathbf{y})\|_2^2 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2$$

will hold for all $\mathbf{x}, \mathbf{y} \in S$ with probability at least p, provided that $m \geq 4\frac{8c(16c+1)}{\epsilon^2}\ln\left(\frac{4|S|^2}{1-p}\right)$.

Proof. The proof of this theorem is left as an exercise to the reader. □

Theorem 3.12.10. *Let $p, \epsilon \in (0,1)$ and Φ as in Theorem 3.12.8. Choose m such that*

$$m \geq s\left(\frac{32c(16c+1)}{\epsilon^2}\right)\ln\left(\frac{\left(\frac{eN}{s}\right)\left(\frac{48}{\epsilon}\right)^2}{(1-p)^{1/s}}\right)$$

*then $\frac{1}{\sqrt{m}}\Phi$ will have the (s,ϵ)-RIP property (see Definition **??**) with probability at least p.*

Proof. The proof of this theorem is left as an exercise to the reader. □

Exercise 3.12.1 (Towards Φ being sparse). *1. $f(x) = p\delta(x) + \frac{(1-p)^{3/2}}{\sqrt{2\pi}}\exp\left(-\frac{x^2(1-p)}{2}\right)$ for $p \in (0,1)$. Show that X with density f is mean 0, variance 1 and sub-gaussian with parameter $c = \frac{1}{2(1-p)}$*

2. Let X have density

$$f(x) = p\delta(x) + \frac{(1-p)}{2}\left[\delta\left(x - \frac{1}{\sqrt{1-p}}\right) + \delta\left(x + \frac{1}{\sqrt{1-p}}\right)\right]$$

for $p \in (0,1)$. Show that X with density f is mean 0, variance 1 and sub-gaussian with parameter $c = \frac{1}{1-p}$

3. Consider Theorem 3.12.7 and how c has to scale in order to end up having fewer non-zero entries in Φ with the same probability decay.

Exercise 3.12.2. *Prove Theorem 3.12.8 (Hint: use union bound) and discuss why it also implies a proof of Theorem 4.1.3.*

Exercise 3.12.3. *Prove that if $\Phi \in \mathbb{R}^{m\times N}$ is an ϵ-JL map of both $S \subset \mathbb{R}^N$ and $T \subset \mathbb{R}^N$ then Φ is an ϵ-JL map of any $R \subset \mathbb{C}^N$ with $\mathrm{Re}\,(R) \subset S$ and $\mathrm{Im}\,(R) \subset T$.*

Exercise 3.12.4. *Prove Theorem 3.12.10. Hint: see Homework 4.2.2 and Theorem 4.3.1 as well as the following consequence of Sterling's approximation*

$$\left(\frac{N}{s}\right)^s \leq \binom{N}{s} \leq \left(\frac{eN}{s}\right)^s, \forall N, s \in \mathbb{Z}^+, N \geq s$$

Exercise 3.12.5 (optional)**.** *Suppose X is a Radamacher random variable. Show that X allows a parameter $c = \frac{1}{2}$ (Definition 3.12.5), i.e. $\mathbb{E}\left[\exp\left(\theta X\right)\right] \leq \exp\left(\frac{\theta^2}{2}\right)$.*

– ADD TO LINEAR ALGEBRA CHAPTER?

Lemma 3.12.11. *If $p > q \geq 1$ then $\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q$, $\forall \mathbf{x} \in \mathbb{R}^n$*

Proof. Consider

$$\begin{aligned}
\|\mathbf{x}\|_p^p &= \sum_{j=1}^n |x_j|^p \\
&= \sum_{j=1}^n |x_j|^q |x_j|^{p-q} \\
&\leq \max_j |x_j|^{p-q} \left(\sum_{j=1}^n |x_j|^q \right)
\end{aligned}$$

Note that

$$\max_{j\in[n]} |x_j|^q \leq \sum_{j=1}^n |x_j|^q \iff \max_{j\in[n]} |x_j|^{p-q} \leq \left(\sum_{j=1}^n |x_j|^q \right)^{\frac{p-q}{q}} = \left(\|\mathbf{x}\|_q^q \right)^{\frac{p-q}{q}}$$

therefore

$$\begin{aligned}
\|\mathbf{x}\|_p^p &\leq \left(\|\mathbf{x}\|_q^q \right)^{\frac{p-q}{q}} \|\mathbf{x}\|_q^q \\
&= \|\mathbf{x}\|_q^p
\end{aligned}$$

which implies the result after taking p-th root □

Chapter 4

A Deterministic Approach to Randomized Numerical Linear Algebra

4.1 Johnson-Lindenstrauss Maps

Definition 4.1.1 (ϵ-JL map)**.** *A matrix $\Phi \in \mathbb{C}^{m\times N}$ is a linear ϵ-JL map of a set $S \subset \mathbb{C}^N$ into $\mathbb{C}^m$ if for all $\mathbf{x} \in S$ $\exists \epsilon_{\mathbf{x}} \in (-\epsilon, \epsilon)$ such that*

$$\|\Phi\mathbf{x}\|_2^2 = (1+\epsilon_{\mathbf{x}})\|\mathbf{x}\|_2^2.$$

Definition 4.1.2 (Set difference)**.** *Let $\tilde{S} \subset \mathbb{C}^N$, then the set difference of $\tilde{S}$ denoted $\tilde{S} - \tilde{S}$ is $\left\{\mathbf{x}-\mathbf{y} | \mathbf{x}, \mathbf{y} \in \tilde{S}\right\} \in \mathbb{C}^N$*

Note. *WTF? When $\epsilon \in (0,1)$, and Φ a ϵ-JL map, then Φ satisfies 1.7 for $x \in \mathcal{F}_{\mathbf{p}} - \mathcal{F}_{\mathbf{p}}$, $X = \ell^2(\mathbb{C}^m)$ and $Y = \ell^2(\mathbb{C}^N)$.*

Perhaps surprisingly, there are simple ways to construct ϵ-JL maps which, other than an upperbound on the cardinality, do not depend on any particular property of S. We will see that by drawing Φ as a random matrix, in a variety of ways, independent of S will result in Φ being a ϵ-JL map for S so long as $m \geq C\log(|S|)$, where C is a (mild) absolute constant.

Theorem 4.1.3. *Let $S \subset \mathbb{C}^N$ be finite. Then there exists a linear ϵ-JL map $\Phi \in \mathbb{C}^{m\times N}$ of the set S into $\mathbb{C}^m$ where $m \leq \frac{C}{\epsilon^2}\log|S|$, and $C \in (0,\infty)$ is an absolute constant independent of both (S and ϵ). Furthermore, Φ may be generated with high probability for any $S \subset \mathbb{C}^N$ given only knowledge of $|S|$, the cardinality of the set.*

For now we will take this theorem as given and work to understand its meaning and consequences.

In order to explain the meaning of the theorem, consider the following two-player game between Alice and Bob. This game will proceed in two phases. In the first phase, Alice selects a finite subset S of the space $\mathbb{C}^N$. The content of S is known only to Alice, however the dimension N of the space and the cardinality $|S|$ is information available to Bob. In the second phase, Alice provides an error bound $\epsilon \in (0,1)$ and Bob must generate an ϵ-JL map Φ for Alice's set S which has at most m rows where $m \leq \frac{C}{\epsilon^2} \log |S|$ rows. Bob wins the game if Φ is an ϵ-JL map of S into $\mathbb{C}^m$, otherwise Alice wins. At first gloss, we may suppose that Alice has the advantage in this game. Afterall, she determines in whatever way she may wish a set S and keeps most of that information secret. Bob has the seemingly more difficult task of mapping a set he knows little about to a lower dimensional space with a distortion error specified by his opponent. To abuse a metaphor, Bob has to draw a faithful, recognizable picture of an object that Alice dreams while he is blindfolded. Shouldn't Alice be able to come up with a set and error for which this is difficult to achieve? Surprisingly, Theorem 4.1.3 states that with high probability Bob will win the game simply by generating a random matrix $\Phi \in \mathbb{C}^{m \times N}$, no matter the set S Alice produces.

Also, note that should $m \geq N$ then the existence of an ϵ-JL map Φ is of little practical use and we could in that case construct isometric embeddings trivially. We are interested in ranges of values ϵ and $|S|$ which lead to compression, i.e. $m \leq N$.

Example 4.1.4 (Generating ϵ-JL maps)**.** *The following random processes generate Φ as in Theorem 4.1.3 with high probability. Note that these are data oblivious maps - they do not depend on any property of the set S other than the cardinality.*

1 Φ *has i.i.d entries where* $\Phi_{(i,j)} \sim \frac{1}{\sqrt{m}} \mathcal{N}(0,1) = \mathcal{N}(0, \frac{1}{m})$

2 Φ *has i.i.d entries where*

$$\Phi_{(i,j)} \sim \begin{cases} \frac{1}{\sqrt{m}} & \text{with probability } 1/2 \\ -\frac{1}{\sqrt{m}} & \text{with probability } 1/2 \end{cases}$$

3 The matrix Φ is a "sparse J-L transform" containing at most $\mathcal{O}(\epsilon^{-1} \log |S|)$ nonzero entries in each column, and thus there are proportionally $\mathcal{O}(\epsilon)$ nonzero entries of Φ. [See[?]].

The above examples have a drawback in that they require $\mathcal{O}(nM)$ memory to store and do not permit a fast matrix-vector multiply - afterall a matrix with independent random values does not have a structure we can exploit to save on storing or in applying to vectors. Our next example will show how we may achieve maps which do have better performance. First however we will need a theorem.

Theorem 4.1.5 (Rahut, Foucart + Krahmer, Ward)**.** *Given the following*

- $U \in \mathbb{C}^{N \times N}$ *a unitary matrix with entries bounded by* $\max_{n,k \in [N]} |U_{k,n}| \leq \frac{K}{\sqrt{N}}$.
- $R \in \mathbb{C}^{m \times N}$ *a matrix obtained by selecting m-rows from the $N \times N$ identity matrix i.i.d. uniformly at random.*

- *$D \in \mathbb{R}^{N\times N}$ be a diagonal matrix with i.i.d ± 1 Rademacher random values on its diagonal.*

Then

$$\sqrt{\frac{N}{m}}RUD$$

Rademacher will be an ϵ-JL map of any given $S \subset \mathbb{R}^N$ into $\mathbb{C}^m$ with probability at least $1-p$ provided that

$$m \geq c\frac{K^2}{\epsilon^2}\log\left(\frac{4|S|}{p}\right)\log^4 N$$

where $c \in \mathbb{R}^+$ is an absolute constant.

add intuitive remark about how random diagonal and unitary matrix spread info out to all rows s.t. you can then safely downsample while preserving info about S. HW assignment seeing what happens when you make U diagonal would be useful. As well as an example S explaining why things can go wrong.

Note that D can be applied in $\mathcal{O}(N)$ time to a vector in $\mathbf{x} \in \mathbb{R}^N$, since it involves scanning the length of the vector and changing signs of some entries. The matrix R can be applied to a vector of length N, $UD\mathbf{x}$ in $\mathcal{O}(N)$-time since it involves scanning the length of the vector and discarding values. Applying U then, since generically it should require at least reading in the inputs should be at least $\mathcal{O}(N)$. Therefore the overall complexity is governed principally by U. If the unitary matrix U admits a fast matrix-vector, like for example using the Fast Fourier transforms to effect a Discrete Fourier transform, then U can be applied to $D\mathbf{x}$ in $\mathcal{O}(N\log N)$-time. In practice, the $\log^4 N$ factor in the bound of m is often ignored with no change in performance.

Corollary 4.1.6. *If U is a DFT then the $\epsilon - JL$ maps from Theorem 4.1.5 have a $\mathcal{O}(N\log N)$ matrix vector multiplication time.*

Example 4.1.7. *Let F be a unitary discrete Fourier transform matrix*

$$F_{n,k} = \frac{1}{\sqrt{N}}e^{\frac{2\pi ink}{N}}$$

If we take $U = F$ and R and D be the matrices described in Theorem 4.1.5 then $\sqrt{\frac{N}{m}}RFD$ is a JL map where

$$\max_{n,k}|F_{n,k}| = \frac{1}{\sqrt{N}}$$

so $K = 1$. The Fast Fourier Transform (FFT) can be used to apply F to any vector $\mathbf{x} \in \mathbb{R}^N$ in $\mathcal{O}(N\log N)$. This is the proto-typical "Fast JL Matrix."

We now consider a direct application of JL-maps to the Nearest Neighbor problem introduced in 17. Recall in the Nearest Neighbor problem we want to find for each element in a set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ another element which is closest with respect to $\|\cdot\|_2$. Here we consider the case where $D \gg \log N$. The naive solution requires $\mathcal{O}(DN^2)$-time. How can we improve on this using a JL map? Note that $S - S$ as in Definition 4.1.2 has the following bound on its cardinality

$$|S - S| \leq N^2 - N + 1.$$

MOVE SET DIFF DEFINITION DOWN HERE CLOSER TO FIRST USE

Let $\Phi \in \mathbb{C}^{m \times D}$ be an ϵ-JL map of $S-S$ with $m \geq \mathcal{O}\left(\frac{\log |S|}{\epsilon^2}\right)$. To improve $NN-$runtime we:

1 Compute $\tilde{S} := \{\Phi\mathbf{x} | \mathbf{x} \in S\}$

2 Run $NN-$algorithm on $\tilde{S}$ instead

Finding the set $\tilde{S}$ takes $\mathcal{O}\left(\frac{ND}{\epsilon^2} \log N\right)$-time and naive nearest neighbors on $\tilde{S}$ takes $\mathcal{O}\left(\frac{N^2}{\epsilon^2} \log N\right)$-time. When $D \gg \log N$, note

$$\frac{N}{\epsilon^2} \log N(D + N) < N^2 D$$

Note that it's possible to combine the JL compression and LSH solution approach to (c, r)-NN for faster speed ups.

Note. *A linear ϵ-JL map Φ of $S \subset \mathbb{C}^N \setminus \{\mathbf{0}\}$ with $\epsilon \in (0, 1)$ must have $S \cap Ker\Phi = \emptyset$. The requirement that the null space of the map is disjoint from the set S can be stated more precisely as uniformly bounding the operator norm on S.*

Definition 4.1.8. *Let $\Phi \in \mathbb{C}^{m \times N}$ and $S \setminus \{0\} \subset \mathbb{C}^N$ be nonempty. The operator norm of Φ on S denoted $\|\Phi\|_{S,2\to 2}$ is defined as*

$$\|\Phi\|_{S,2\to 2} = \sup_{\mathbf{x} \in S \setminus \{0\}} \frac{\|\Phi\mathbf{x}\|_2}{\|\mathbf{x}\|_2}.$$

Lemma 4.1.9. *$\|\Phi\|_{S,2\to 2}$ is a semi-norm if $S \setminus \{0\} \neq \emptyset$ and it is a norm if S contains N linearly independent vectors.*

Lemma 4.1.10. *$\|\Phi\|_{S,2\to 2} = \|\Phi\|_{2\to 2}$ if $\left\{\frac{\mathbf{y}}{\|\mathbf{y}\|_2} | \mathbf{y} \in S \setminus \{0\}\right\} = \overline{B(0,1)} \setminus B(0,1) = \delta B(0,1)$*

Lemma 4.1.11. *If Φ is an ϵ-JL map of S then the following inequalities hold*

$$\|\Phi\|_{S,2\to 2} \leq \sqrt{1+\epsilon}, \inf_{\mathbf{y} \in S \setminus \{0\}} \frac{\|\Phi\mathbf{y}\|_2}{\|\mathbf{y}\|_2} \geq \sqrt{1-\epsilon}, \sup_{\mathbf{y} \in S \setminus \{0\}} \frac{|\langle (\Phi^*\Phi - I)\mathbf{y}, \mathbf{y}\rangle|}{\|\mathbf{y}\|_2^2} \leq \epsilon$$

Norm preserving maps of certain sets which are geometrically related to S can also preserve the geometry of S itself.

CMSE 890 Lecture 11

Motivation! Preserve all Euclidean geometry of arbitrary set in low-dim with random matrix is funny, maybe unbelievable

Introduce polarization identity earlier (Chap 1)? Figure!?!

Lemma 4.1.12. *Let $S \subset \mathbb{C}^N$ and $\epsilon \in (0,1)$. If $\Phi \in \mathbb{C}^{m\times N}$ is an ϵ-JL map of S' into $\mathbb{C}^m$, where*

$$S' = \left\{ \frac{\mathbf{x}}{\|\mathbf{x}\|_2} + \frac{\mathbf{y}}{\|\mathbf{y}\|_2}, \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2}, \frac{\mathbf{x}}{\|\mathbf{x}\|_2} + i\frac{\mathbf{y}}{\|\mathbf{y}\|_2}, \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - i\frac{\mathbf{y}}{\|\mathbf{y}\|_2} | \mathbf{x}, \mathbf{y} \in S\backslash\{0\} \right\}$$

it will satisfy $\forall \mathbf{x}, \mathbf{y} \in S$

$$|\langle \Phi\mathbf{x}, \Phi\mathbf{y}\rangle - \langle \mathbf{x}, \mathbf{y}\rangle| \leq 4\epsilon \|\mathbf{x}\|_2 \|\mathbf{y}\|_2. \tag{4.1}$$

Proof. Consider the case where $\mathbf{x} = \mathbf{0}$ or $\mathbf{y} = \mathbf{0}$ then the inequality holds because $0 \leq 0$. So next we suppose $\mathbf{x}, \mathbf{y} \neq 0$. Consider the normalizations $\mathbf{u} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}, \mathbf{v} = \frac{\mathbf{y}}{\|\mathbf{y}\|_2}$. The polarization identity relates inner products with norms. Observe,

$$\begin{aligned}
|\langle \Phi\mathbf{u}, \Phi\mathbf{v}\rangle - \langle \mathbf{u}, \mathbf{v}\rangle| &= \left| \frac{1}{4} \sum_{\ell=0}^{3} i^\ell \left(\|\Phi\mathbf{u} + i^\ell \Phi\mathbf{v}\|_2^2 - \|\mathbf{u} + i^\ell \mathbf{v}\|_2^2 \right) \right| \\
&= \left| \frac{1}{4} \sum_{\ell=0}^{3} i^\ell \epsilon_{\mathbf{u},\mathbf{v},\ell} \|\mathbf{u} + i^\ell \mathbf{v}\|_2^2 \right| \\
&\leq \frac{1}{4} \sum_{\ell=0}^{3} \epsilon \left(\|\mathbf{u}\|_2 + \|\mathbf{v}\|_2 \right)^2 \\
&= 4\epsilon.
\end{aligned}$$

□

Note. *If $\Phi \in \mathbb{C}^{m\times N}$ has ± 1 Radamaecher entries, $m \geq \frac{c}{\epsilon^2} \log\left(2|S|^2\right)$ will work w.h.p.*
If $M = RFD$ "fast $\epsilon-$JL map" then $m \geq \frac{c}{\epsilon^2} \log\left(2|S|^2\right) \cdot \log^4 N$ will work w.h.p.

Note. *Constructing Φ still only depends on the original information about the cardinality of S since $|S'| \leq 4|S|^2$, and so we can apply Theorem 4.1.3 whether we have $|S|$ or $|S'|$.*

Note. *Lemma 4.1.12 and Theorem 4.1.3 imply that $\exists \Phi \in \mathbb{C}^{m\times N}$ a ϵ-JL map where the inequality 4.1 holds for any choice of S provided that*

$$m = \frac{c}{\epsilon^2} \log\left(4|S|^2\right)$$

Should we wish to construct a ϵ-JL map with fast matrix-vector multiply, we can use an RFD matrix like that in Example 4.1.7, and our sketching dimension m is

$$m = \frac{c}{\epsilon^2}\log\left(4|S|^2\right)\log^4 N$$

We can use 4.1.12 to implement a fast approximate matrix multiplication algorithm. We will show how to use any type of ϵ-JL map to achieve the speed-up.

Lemma 4.1.13. *Let $V \in \mathbb{C}^{N\times p}$ and $U \in \mathbb{C}^{N\times q}$ have unit ℓ^2-normalized columns. Suppose that $\Phi \in \mathbb{C}^{m\times N}$ satisfies Equation 4.1 from Lemma 4.1.12 where Complaint about notation in def. of S $S = \{\mathbf{u}_j|\mathbf{u}_j = U[:,j]\} \cup \{\mathbf{v}_j|\mathbf{v}_j = V[:,j]\}$. Then*

$$\left|(V^*\Phi^*\Phi U - V^*U)_{k,j}\right| \leq 4\epsilon,\ \forall k \in [p], \forall j \in [q]$$

Proof. Note that $|S| = p + q$ thus $S' \leq 4(p+q)^2$ from Lemma 4.1.12. Furthermore note that

$$\Phi V = \begin{pmatrix} | & | & \cdots & | \\ \Phi\mathbf{v}_1 & \Phi\mathbf{v}_2 & \cdots & \Phi\mathbf{v}_p \\ | & | & \cdots & | \end{pmatrix},\ \Phi U = \begin{pmatrix} | & | & \cdots & | \\ \Phi\mathbf{u}_1 & \Phi\mathbf{u}_2 & \cdots & \Phi\mathbf{u}_q \\ | & | & \cdots & | \end{pmatrix}$$

So note then that $\left((\Phi V)^*\Phi U\right)_{k,j} = \langle\Phi\mathbf{v}_k, \Phi\mathbf{u}_j\rangle$. Therefore for all choices of k, j and given Lemma 4.1.12 we have

$$\begin{aligned}\left|(V^*\Phi^*\Phi U - V^*U)_{k,j}\right| &= |\langle\Phi\mathbf{v}_k, \Phi\mathbf{u}_j\rangle - \langle\mathbf{v}_k, \mathbf{u}_j\rangle| \\ &\leq 4\epsilon\|\mathbf{v}_k\|_2\|\mathbf{u}_j\|_2 \\ &= 4\epsilon\end{aligned}$$

□

Notes are structured funky.

Note. *We can have $\Phi \in \mathbb{C}^{m\times N}$:*

- *with Rademacher entries with $m \sim \frac{c\log(p+q)}{\epsilon^2}$*

a fast RFD with $m \sim \frac{c\log(p+q)\cdot\log^4 N}{\epsilon^2}$

Note that we know there exists $\Phi \in \mathbb{C}^{m\times N}$ that satisfies the needed inequality from Lemma 4.1.12 such that

$$m = \mathcal{O}\left(\epsilon^{-2}\log\left(\max(p,q)^2\right)\right)$$

Theorem 4.1.14 (Fast Matrix-Matrix Multiply)**.** *Let $A \in \mathbb{C}^{p\times N}$ and $B \in \mathbb{C}^{N\times q}$ have SVDs given by $A = U_1\Sigma_1V^*$ and $B = U\Sigma_2V_2^*$ and suppose that $\Phi \in \mathbb{C}^{m\times N}$ satisfies the conditions of Lemma 4.1.13 for U and V. Then*

$$\|A\Phi^*\Phi B - AB\|_F \leq 4\epsilon\|A\|_F\|B\|_F$$

Proof. We will expand the quantity of interest according the SVD of the factors A and B

$$\begin{aligned}
\|A\Phi^*\Phi B - AB\|_F &= \|U_1\Sigma_1 V^*\Phi^*\Phi U\Sigma_2 V_2^* - U_1\Sigma_1 V^* U\Sigma_2 V_2^*\|_F \\
&= \|U_1\Sigma_1\left(V^*\Phi^*\Phi U - V^*U\right)\Sigma_2 V_2^*\|_F \\
&= \|\Sigma_1\left(V^*\Phi^*\Phi U - V^*U\right)\Sigma_2\|_F \\
&= \sqrt{\sum_{k=1}^{p}\sum_{j=1}^{q}(\Sigma_1)_{k,k}^2\left|V^*\Phi^*\Phi U - V^*U\right|_{k,j}^2(\Sigma_2)_{j,j}^2} \\
&\leq \sqrt{\sum_{k=1}^{p}\sum_{j=1}^{q}\sigma_k(A)^2(4\epsilon)^2\sigma_j(B)^2} \\
&= 4\epsilon\sqrt{\sum_{k=1}^{p}\sigma_k(A)^2}\sqrt{\sum_{j=1}^{q}\sigma_j(B)^2} \\
&= 4\epsilon\|A\|_F\|B\|_F
\end{aligned}$$

□

What are the savings in runtime then if we wish to approximate matrix-matrix multiplication in this way? To simplify the comparison suppose $p, q = N$ (or at comparable at any rate). Usual matrix multiplication then consists of computing N^2 entries, each consisting of the inner product of two N dimensional vectors, i.e. $\mathcal{O}(N^3)$. If we use Theorem 4.1.14 then there are three major operations to consider

1. Compute the product ΦB which takes $\mathcal{O}(mN^2)$
2. Compute the product $\Phi A*$ which takes $\mathcal{O}(mN^2)$. Conjugate transposition takes at most $\mathcal{O}(N^2)$ operations
3. Compute $\left(A\Phi^*\right)\left(\Phi B\right)$ which takes $\mathcal{O}(nN^2)$

We have seen from Lemma 4.1.12 and Theorem 4.1.3 that $m = \mathcal{O}\left(\epsilon^{-2}\log N\right)$. So the total runtime for the approximate matrix-matrix multiplication is $\mathcal{O}(\frac{N^2}{\epsilon^2}\log N)$ vs $\mathcal{O}(N^3)$ for naive matrix multiplication.

Lemma 4.1.15. *Let $S \subset \mathbb{R}^N$ and $\epsilon \in (0,1)$, then an ϵ-JL map $\Phi \in \mathbb{C}^{m\times N}$ of the set*

$$S' = \left\{\frac{\mathbf{x}}{\|\mathbf{x}\|_2} + \frac{\mathbf{y}}{\|\mathbf{y}\|_2}, \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2}\middle| \mathbf{x},\mathbf{y} \in S\right\}$$

will satisfy $\forall \mathbf{x},\mathbf{y} \in S$

$$\left|\mathrm{Re}\left(\langle\Phi\mathbf{x},\Phi\mathbf{y}\rangle\right) - \langle\mathbf{x},\mathbf{y}\rangle\right| \leq \epsilon\|\mathbf{x}\|_2\|\mathbf{y}\|_2$$

Up to this point, Theorem 4.1.3 and the subsequent discussion has dealt with finite sets of points S. We now turn to the question of whether these results can be applied to infinite sets. We will begin by building new from old; infinite sets which are constructed from finite ones.

Definition 4.1.16 (Cones)**.** *The conical region generated by $S \subset \mathbb{C}^N$ is*

$$cone(S) = \{\alpha x | x \in S, \alpha \in \mathbb{C}\}$$

As an immediate consequence of the linearity, an ϵ-JL map Φ of set S, will be an ϵ-JL map of cone(S) and vice-versa.

Another infinite set of importance is the convex hull of a set of points.

Definition 4.1.17 (Convex Hulls)**.** *The convex hull of a $S \subset \mathbb{C}^N$ is*

$$conv(S) = \bigcup_{j=1}^{\infty} \left\{ \sum_{\ell=1}^{j} \alpha_\ell \mathbf{x}_\ell | x_1, \ldots, x_j \in S,\, \alpha_1, \ldots, \alpha_j \in [0,1] s.t. \sum_{\ell=0}^{N} \alpha_\ell = 1 \right\}$$

We have in this next theorem that the infinitude of points in the convex hull can always be reduced to a finite number of points from the original set. That is that each point in conv(S) where $S \subset \mathbb{R}^N$ can be expressed as a convex combination of at most $N+1$ point from S.

Theorem 4.1.18 (Caratheadory)**.** *Given $S \subset \mathbb{R}^N$, $\forall \mathbf{x} \in conv(S)$, $\exists \mathbf{y}_1, \ldots, \mathbf{y}_{\tilde{N}} \subseteq S$?!?, $\tilde{N} = \min(|S|, N+1)$, such that $\mathbf{x} = \sum_{\ell=1}^{\tilde{N}} \alpha_\ell \mathbf{y}_\ell$ for some $\alpha_1, \ldots, \alpha_{\tilde{N}} \in [0,1]$, $\sum_{\ell=1}^{\tilde{N}} \alpha_\ell = 1$.*

Theorem 4.1.19. *Suppose $S \subset \overline{B_2(\mathbf{0}, \gamma)} \subset \mathbb{R}^N$ and $\epsilon \in (0,1)$. Let $\Phi \in \mathbb{C}^{m \times N}$ be an $\left(\frac{\epsilon}{4\gamma^2}\right)$-JL map of the set S' defined as in Lemma 4.1.15, then*

$$|\langle \Phi\mathbf{x}, \Phi\mathbf{y} \rangle - \langle \mathbf{x}, \mathbf{y} \rangle| \leq \epsilon \tag{4.2}$$

$\forall \mathbf{x}, \mathbf{y} \in conv(S)$

Proof. Let $\mathbf{x}, \mathbf{y} \in \text{conv}(S)$. By Theorem 4.1.18, $\exists \{\mathbf{y}_i\}_{i=1}^{\tilde{N}}$, $\{\mathbf{x}_i\}_{i=1}^{\tilde{N}} \subset S$, $\{\alpha_\ell\}_{\ell=1}^{\tilde{N}}, \{\beta_\ell\}_{\ell=1}^{\tilde{N}} \subset [0,1]$ such that

$$\mathbf{x} = \sum_{\ell=1}^{\tilde{N}} \alpha_\ell \mathbf{x}_\ell,\ \mathbf{y} = \sum_{\ell=1}^{\tilde{N}} \beta_\ell \mathbf{y}_\ell$$

So,

$$\begin{aligned}
|\langle \Phi\mathbf{x}, \Phi\mathbf{y}\rangle - \langle \mathbf{x}, \mathbf{y}\rangle| &= \left| \sum_{\ell=1}^{\tilde{N}} \sum_{j=1}^{\tilde{N}} \alpha_\ell \beta_j \left(\langle \Phi\mathbf{x}_\ell, \Phi\mathbf{y}_j\rangle - \langle \mathbf{x}_\ell, \mathbf{y}_j\rangle \right) \right| \\
&\leq 4 \sum_{\ell=1}^{\tilde{N}} \sum_{j=1}^{\tilde{N}} \alpha_\ell \beta_j \left(\frac{\epsilon}{4\gamma^2} \right) \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \\
&\leq \epsilon \left(\sum_{\ell=1}^{\tilde{N}} \alpha_\ell \right) \left(\sum_{j=1}^{\tilde{N}} \beta_j \right) \\
&= \epsilon
\end{aligned}$$

where we have used the embedding error $\left(\frac{\epsilon}{4\gamma^2}\right)$ and the fact that all norms of vectors in this case will be less than γ □

Corollary 4.1.20. *If* $\inf_{x \in conv(S)} \|x\|_2 \geq 1$ *then* Φ *as in Theorem 4.1.19 will also be an* ϵ*-JL map of* $conv(S)$ *into* $\mathbb{C}^m$

Proof. Consider 4.2 with $\mathbf{x} = \mathbf{y}$ to obtain

$$\left| \|\Phi\mathbf{x}\|_2^2 - \|\mathbf{x}\|_2^2 \right| \leq \epsilon \leq \epsilon \|\mathbf{x}\|_2^2$$

which is the ϵ-JL property. □

For points near zero, for example any points in the intersection Always use ℓ^2? No need for extra subscript? $S \cap B_{\ell^2}(\mathbf{0}, \epsilon)$, Theorem 4.1.19 does not provide useful relative errors bounds. This occurs because the left hand side of 4.2 can be very small for these vectors near zero relative to the fixed ϵ.

Recall, S is finite and (usually) $\text{conv}(S)$ is infinite.

However, with the Corollary 4.1.20, when $\mathbf{x} = \mathbf{y}$ and $x \in \text{conv}(S) \setminus B_{\ell^2}(\mathbf{0}, \alpha)$ then we can achieve an ϵ-JL embedding MAP? Embedding versus map needs to be corrected... of the convex hull less the ball about zero, $\text{conv}(S) \setminus B_{\ell^2}(\mathbf{0}, \alpha)$, by applying a $\left(\frac{\epsilon}{\alpha^2}\right)$-JL to S.

Exercise 4.1.1. *If* B *is the closed unit ball for any norm on* $\mathbb{C}^N$*, show that* $B - B$ *is the ball of radius 2.*

Exercise 4.1.2. *Prove lemma 4.1.9*

Exercise 4.1.3. *Prove lemma 4.1.10*

Exercise 4.1.4. *Prove lemma 4.1.11*

Exercise 4.1.5. *Prove lemma 4.1.15*

Exercise 4.1.6. *Show that Theorem 4.1.19 still holds if* $S \in \overline{B_{\ell^2}(\mathbf{0}, \gamma)} \subset \mathbb{C}^N$.

Exercise 4.1.7. *Can just call* $R = T \cup S$ *and then show that it JL's* $R \times R$. *Wording confusing. Let* $A \in \mathbb{C}^{m/2 \times N/2}$ *be an* ϵ*-JL map of* $T \cup S \subset \mathbb{C}^{N/2}$. *Prove that for* $\mathbf{x}_1, \mathbf{x}_2 \in S \cup T$, $g : \mathbb{C}^N \to \mathbb{C}^m$, *defined by* $g(\mathbf{x}_1, \mathbf{x}_2) = (A\mathbf{x_1}, A\mathbf{x_2})$ *is an* ϵ*-JL embedding of* $(S \times T) \cup (T \times S) \cup (S \times S) \cup (T \times T)$

Exercise 4.1.8. *Wording confusing. Fix* $\epsilon \in (0, 1)$ *and let* $A \in \mathbb{C}^{\tilde{m} \times N}$ *be a* ϵ*-JL map of* $(S - S) \cup S$ *and* $G \in \mathbb{C}^{m \times \tilde{m}}$ *be an* ϵ*-JL map of* $A(S) \subset \mathbb{C}^{\tilde{m}}$, $S \subset \mathbb{C}^N$ *then*

1. $|A(S)| = |S|$
2. GA *is a* 3ϵ*-JL embedding of* S *into* $\mathbb{C}^m$

4.2 Covering Numbers of Balls (MTH 994 Lecture 3)

Next we turn to covering numbers, which will enable us to apply ϵ-JL maps to more general infinite sets. We begin then with these definitions.

Definition 4.2.1 (δ-cover)**.** *Let* $T \subseteq \mathbb{C}^N$. *A* δ*-cover of* T *with respect to norm* $\|\cdot\|_X$ *is a subset* $S \subseteq T$ *such that* $\forall \mathbf{x} \in T$, $\exists \mathbf{y} \in S$ *with* $\|\mathbf{x} - \mathbf{y}\|_X < \delta$. *Hence,*

$$T \subseteq \bigcup_{\mathbf{y} \in S} B_X(\mathbf{y}, \delta).$$

Note that $B_X(\mathbf{y}, \delta)$ is the open ball with center $\mathbf{y} \in \mathbb{C}^N$ and radius δ with respect to the norm $\|\cdot\|_X$. Usually it will be clear from context the space and norm, and so we'll simplify notation and write instead $B(\mathbf{y}, \delta)$

Definition 4.2.2 (δ-covering Number)**.** *The* δ*-covering number of* $T \subseteq \mathbb{C}^N$ *with respect to* $\|\cdot\|_X$, *denoted* $C_\delta^X(T)$, *is the smallest integer such that there exists a* δ*-cover* $S \subseteq T$ *where* $|S| = C_\delta^X(T)$. *If no such integer exists we say that* $C_\delta^X(T) = \infty$

Definition 4.2.3 (δ-packing)**.** *Let* $T \subseteq \mathbb{C}^N$. *A* δ*-packing of* T *with respect to norm* $\|\cdot\|_X$ *is a subset* $S \subseteq T$ *such that* $\forall \mathbf{x}, \mathbf{y} \in S$ *with* $\mathbf{x} \neq \mathbf{y}$,

$$B_X(\mathbf{x}, \delta/2) \cap B_X(\mathbf{y}, \delta/2) = \emptyset.$$

Definition 4.2.4 (δ-packing Number)**.** *The* δ*-packing number of* $T \subseteq \mathbb{C}^N$ *with respect to* $\|\cdot\|_X$, *denoted* $P_\delta^X(T)$, *is the largest integer such that there exists a* δ*-packing* $S \subseteq T$ *where* $|S| = P_\delta^X(T)$. *If no such integer exists we say that* $P_\delta^X(T) = \infty$.

Lemma 4.2.5. *Let* $T \subseteq \mathbb{C}^N$ *and* $\delta \in (0, \infty)$. *Then*

$$P_{2\delta}^X(T) \leq C_\delta^X(T) \leq P_\delta^X(T)$$

Proof. Bad notation conflict for packing versus packing number? Let $P_{2\delta} \subset T$ be a maximal 2δ-packing of T and $C_\delta \subseteq T$ be a minimal δ-cover of T. Each point $\mathbf{x} \in P_{2\delta}$ is closest to a different point $\mathbf{y} \in C_\delta$. To see this, suppose to the contrary that for $\mathbf{x}_1, \mathbf{x}_2, \in P_{2\delta}, \mathbf{x}_1 \neq \mathbf{x}_2$ there was some point $\mathbf{y} \in C_\delta$ such that $\mathbf{x}_1, \mathbf{x}_2 \in B(\mathbf{y}, \delta)$ this implies that $\mathbf{y} \in B_X(\mathbf{x}_1, \delta) \cap B_X(\mathbf{x}_2, \delta)$ which is a contradiction.

So, since each point in P_δ can be identified with at least one point in C_δ. We thus define an injection $f : P_{2\delta} \to C_\delta$ where $f(\mathbf{x}) = \mathbf{y}$, $x \in B(\mathbf{y}, \delta)$. Since f is an injection, we have that the cardinality of C_δ must be equal to or larger than $P_{2\delta}$, which is equivalent to the left hand side of the desired inequality.

Next, suppose P_δ is a maximal δ-packing of T. Now suppose for eventual contradiction that there exists a point $\mathbf{y} \in T, \mathbf{y} \notin P_\delta$ such that $\|\mathbf{x} - \mathbf{y}\| \geq \delta$, $\forall x \in P_\delta$. This implies that $B_X(\mathbf{x}, \delta/2) \cap B_X(\mathbf{y}, \delta/2) =$. Thus $P_\delta \cup \{\mathbf{y}\}$ is a δ-packing of T. This contradicts that P_δ is maximal. So, for all points $\mathbf{y} \in T$, there is $\mathbf{x} \in P_\delta$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \delta$, which is to say P_δ is a δ-covering of T, and therefore the cardinality of P_δ is equal to or larger than the δ-covering number for T. This is the right hand side of the desired inequality. □

Lemma 4.2.6. *Let $T \subseteq \mathbb{R}^N$ and $\delta \in (0, \infty)$. Furthermore let B denote the unit ball $B_X(0, 1)$ in $\mathbb{R}^N$ with respect to some norm $\|\cdot\|_X$. Then*

$$\left(\frac{1}{\delta}\right)^N \frac{Vol(T)}{Vol(B)} \leq C_\delta^X(T) \leq P_\delta^X(T) \leq \left(\frac{2}{\delta}\right)^N \frac{Vol\left(T + \left(\frac{\delta}{2}\right) B\right)}{Vol(B)}$$

holds, where $Vol(T) = \int_T 1 dV$, the Lebesgue measure of T in $\mathbb{R}^N$.

Has $T + S = T - (-S) = \{t + s | \forall t \in T, s \in S\}$ been defined formally yet? Just use regular Euclidean balls below and scrap the generality?

Proof. Suppose C_δ is a minimal δ cover of T. By definition then of δ-cover

$$T \subseteq \bigcup_{\mathbf{y} \in C_\delta} B(\mathbf{y}, \delta)$$

So, using properties of volume – and then mention measure theory stuff in footnote "sub-additivity of measurable sets, translation invariance, and scaling" we have

$\text{Vol}(T) \leq \text{Vol}\left(\bigcup_{\mathbf{y} \in C_\delta} B(\mathbf{y}, \delta)\right) \leq C_\delta^X \text{Vol}\left(B(\mathbf{y}, \delta)\right) = C_\delta^X \delta^N \text{Vol}\left(B(0, 1)\right)$

Rearranging terms, we obtain the left hand side of the desired inequality

$$\left(\frac{1}{\delta}\right)^N \frac{\text{Vol}\left(T\right)}{\text{Vol}\left(B\right)} \leq C_\delta^X(T)$$

Now suppose P_δ is a maximal δ-packing of T. It follows that

$$\bigcup_{\mathbf{y} \in P_\delta} B(\mathbf{y}, \delta/2) \subset T + B(0, \delta/2)$$

Since the balls that make up the δ-packing of T are disjoint, we have that their measure is additive. Again, using translation invariance and scaling, this implies

$$\mathrm{Vol}\left(\bigcup_{\mathbf{y}\in P_\delta} B(\mathbf{y},\delta/2)\right) = P_\delta^X(T)\left(\frac{\delta}{2}\right)^N \mathrm{Vol}\left(B(\mathbf{0},1)\right) \leq \mathrm{Vol}(T + B(0,\delta/2))$$

Which after rearranging terms matches the right hand side of the desired inequality $\square$

Corollary 4.2.7. $\left(\frac{1}{\delta}\right)^N \leq C_\delta^X(B) \leq \left(1+\frac{2}{\delta}\right)^N$ *for all norms* $\|\cdot\|_X$ *on* $\mathbb{R}^N$

Proof. We can apply lemma 4.2.6 where $T = B(0,1)$. So,

$$\left(\frac{1}{\delta}\right)^N \frac{\cancel{\mathrm{Vol}(B)}}{\cancel{\mathrm{Vol}(B)}} \leq C_\delta^X(T)$$

yields the first half of the corollary. Next, note that using $T = B(0,1)$ we see that

$$B(0,1) + B\left(0,\frac{\delta}{2}\right) \subseteq B\left(0,\left(1+\frac{\delta}{2}\right)\right)$$

Note that by scaling, we have the following for the volume calculation

$$\mathrm{Vol}\left(B\left(0,\left(1+\frac{\delta}{2}\right)\right)\right) = \left(1+\frac{\delta}{2}\right)^N \mathrm{Vol}(B(0,1))$$

Putting this into the previous lemma, we see

$$C_\delta^X(T) \leq \left(\frac{2}{\delta}\right)^N \left(1+\frac{\delta}{2}\right)^N \frac{\cancel{\mathrm{Vol}(B(0,1))}}{\cancel{\mathrm{Vol}(B)}} = \left(1+\frac{2}{\delta}\right)^N$$

Which completes the second inequality $\square$

Note that if we add the assumption in the corollary that $\delta \in (0,1)$ then we can bound $\left(1+\frac{2}{\delta}\right) \leq \left(\frac{1}{\delta}+\frac{2}{\delta}\right) = \frac{3}{\delta}$, for a more concise, though less tight bound.

Corollary 4.2.8. *If* $S \subseteq \overline{B_X(0,1)} \subset \mathbb{R}^N$ *then*

$$C_\delta^X(S) \leq \left(1+\frac{2}{\delta}\right)^N.$$

Proof. By observing

$$\mathrm{Vol}\left(S + B\left(\mathbf{0},\frac{\delta}{2}\right)\right) \leq \mathrm{Vol}\left(\overline{B_X(0,1)} + \overline{B\left(\mathbf{0},\frac{\delta}{2}\right)}\right)$$

and applying the same reasoning as corollary 4.2.7, we achieve the result. $\square$

Exercise 4.2.1. *Consider the identification of $\mathbb{C}^N$ with $\mathbb{R}^{2N}$ given by the map*

$$f : \begin{pmatrix} x_1 \\ \vdots \\ x_{2N} \end{pmatrix} \to \begin{pmatrix} x_1 + ix_2 \\ \vdots \\ x_{2N-1} + ix_{2N} \end{pmatrix}$$

1. *Verify that $f : \mathbb{R}^{2N} \to \mathbb{C}^N$ is a bijection with Why is c here. Just define with $c = 1$ since it's much easier...*

$$f^{-1}(c\mathbf{z}) = \begin{pmatrix} \mathrm{Re}(c)\mathrm{Re}(z_1) - \mathrm{Im}(c)\mathrm{Im}(z_1) \\ \mathrm{Re}(c)\mathrm{Im}(z_1) + \mathrm{Im}(c)\mathrm{Re}(z_1) \\ \vdots \end{pmatrix}, \forall c \in \mathbb{C}, \mathbf{z} \in \mathbb{C}^N$$

2. *Show that $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ and $f(c\mathbf{x}) = cf(\mathbf{x})$ both hold $\forall \mathbf{x}, \mathbf{y}\mathbb{R}^{2N}$ and $c \in \mathbb{R}$*
3. *Verify that $\|f(\mathbf{x})\|_2 = \|\mathbf{x}\|_2$, $\forall \mathbf{x} \in \mathbb{R}^{2N}$, i.e. f is an isometry*
4. *Let $\|\cdot\|_X$ be a norm on $\mathbb{C}^N$ over $\mathbb{C}$. Prove that $\|\cdot\|_{X'} : \mathbb{R}^{2N} \to [0,\infty)$ defined by $\|\cdot\|_{X'} = \|f(\mathbf{x})\|_X$ is a norm on $\mathbb{R}^{2N}$ over R.*
5. *Prove that $f\left(B_{X'}(\mathbf{y}, r)\right) = B_X\left(f(\mathbf{y}), r\right)$ for all $\mathbf{y} \in \mathbb{R}^{2N}$ and $r \in [0,\infty)$*

Exercise 4.2.2. *For $T \subset \mathbb{C}^N$ we have $Vol(T) = Vol(f^{-1}(T))$. Modify the proofs of Lemma 4.2.6 and Corollary 4.2.7 to prove that*

$$\left(\frac{1}{\delta}\right)^{2N} \leq C_\delta^X\left(B_X\right) \leq \left(\frac{3}{\delta}\right)^{2N}$$

for all norms $\|\cdot\|_X$ on $\mathbb{C}^N$ and $\delta \in (0,1)$

4.3 Linear Subspace Embedding (CMSE 890 Lecture 12)

Next we turn to covering numbers, which will enable us to apply ϵ-JL maps to more general infinite sets. We begin then with these definitions.

4.3.1 Unit Balls in r-dim Subspaces of $\mathbb{C}^N$

1 $\mathcal{L}$ is an $r-$dimensional subspace of $\mathbb{C}^N$, then the unit ball in $\mathcal{L}$ is $\mathcal{L} \bigcap B_2^N$, where $B_2^N := \{x \in \mathbb{C}^N \big| \|x\|_2 \leq 1.\}$

2 Given an r-dimensional linear subspace of some orthonormal basis $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$ as

$$\mathcal{L}_{\mathcal{B}}^r = \{a\mathbf{x} | \alpha \in \mathbb{C}, \mathbf{x} \in S_{\mathcal{B}}^r\} \subset \mathbb{C}^N$$

Strange and annoying way to define this subspace. Where $S^r_{\mathcal{B}}$ denotes the r-dimensional unit sphere with respect to the basis $\mathcal{B}$

$$S^r_{\mathcal{B}} = \left\{ \mathbf{x} \in \mathbb{C}^N | \mathbf{x} = \sum_{j=1}^{r} c_j \mathbf{b}_j, \mathbf{c} \in \mathbb{C}^r \text{ s.t. } \|\mathbf{c}\|_2 = 1 \right\}$$

Note that it is possible to represent any r-dimensional subspace as some $\mathcal{L}^r_{\mathcal{B}}$. Our strategy for proving the main result of this section will be then to embed a sufficiently dense cover of $S^r_{\mathcal{B}} \subset \mathbb{C}^N$

Theorem 4.3.1 (Subspace Embeddings)**.** *Fix $\epsilon \in (0,1)$. Let $\mathcal{L}^r_{\mathcal{B}} \subset \mathbb{C}^N$ be a r-dimensional subspace of $\mathbb{C}^N$ with respect to some orthonormal basis $\mathcal{B}$ and furthermore let $C \subset S^r_{\mathcal{B}}$ be a minimal $\left(\frac{\epsilon}{16}\right)$-cover of $S^r_{\mathcal{B}} \subset \mathcal{L}^r_{\mathcal{B}}$. Then if $\Phi \in \mathbb{C}^{m \times N}$ is an $\left(\frac{\epsilon}{2}\right)$-JL map of C into $\mathbb{C}^m$ it will also satisfy*

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \|\Phi\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{x}\|_2^2, \forall \mathbf{x} \in \mathcal{L}^r_{\mathcal{B}} \tag{4.3}$$

Proof. improve this proof.By linearity of Φ it suffices to prove that :

$$(1-\epsilon) \leq \|\Phi x\|_2^2 \leq (1+\epsilon) \quad \forall x \in S^r_{\mathcal{B}}.$$

Note that $S^r_{\mathcal{B}}$ is a compact set, and therefore will contain its maximal element $\mathbf{x}$. That is, $\exists \mathbf{x} \in S^r_{\mathcal{B}}$ such that

$$\|\Phi\mathbf{x}\|_2 = \|\Phi\|_{S^r_{\mathcal{B}}, 2\to 2} = \|\Phi\|_{\mathcal{L}^r_{\mathcal{B}}} = \gamma$$

Let $\mathbf{y} \in C$ be such that $\|\mathbf{x}-\mathbf{y}\|_2 < \epsilon/16$. Then, after noting that $\mathbf{x}$ and $\mathbf{y}$ both are of unit norm, we have by the triangle inequality and $\left(\frac{\epsilon}{2}\right)$-JL property of Φ:

$$\begin{aligned}
\gamma - 1 &= \|\Phi\mathbf{x}\|_2 - \|\mathbf{x}\|_2 \\
&\leq \|\Phi\mathbf{y}\|_2 + \|\Phi(\mathbf{x}-\mathbf{y})\|_2 - \|\mathbf{x}\|_2 \\
&\leq \left(1+\frac{\epsilon}{2}\right)^{1/2} \|\mathbf{y}\|_2 + \frac{\epsilon}{2}\gamma - 1
\end{aligned}$$

After rearranging terms and using the inequality $(a+b)^2 \geq a^2 + b^2$ and the fact that $\epsilon^2 < \epsilon$, we have

$$\gamma \leq \frac{1+\epsilon/4}{1-\epsilon/16} = 1 + \epsilon/3$$

That is, we have shown that $\|\Phi\mathbf{x}\|_2^2 < 1+\epsilon$, $\forall x \in S^r_{\mathcal{B}}$. This establishes the required upper bound for our desired result.

For the lower bound, let $\beta = \inf_{\mathbf{z} \in S^r_{\mathcal{B}} \setminus \{0\}} \|\Phi\mathbf{z}\|_2$. The infimum is included in the set since the set is compact and the function continuous. Thus there exists $\mathbf{x} \in S^r_{\mathcal{B}}$, with $\|\Phi\mathbf{x}\|_2 = \beta$.

Now we take a point in the cover, $\mathbf{y} \in C$. So $\|\mathbf{x} - \mathbf{y}\|_2 < \epsilon/16$. Now we use the reverse triangle inequality to observe:

$$\begin{aligned}
\beta - 1 = \|\Phi\mathbf{x}\|_2 - 1 &\geq \|\Phi\mathbf{y}\|_2 - \|\Phi(\mathbf{x}-\mathbf{y})\|_2 - 1 \\
&\geq \left(1 - \frac{\epsilon}{2}\right)^{1/2} \|\mathbf{y}\|_2 - \gamma\left(\frac{\epsilon}{16}\right) - 1 \\
&\geq \left(1 - \frac{\epsilon}{2}\right) - \left(1 + \frac{\epsilon}{3}\right)\left(\frac{\epsilon}{16}\right) - 1 \\
&\geq \left(1 - \frac{\epsilon}{3}\right) - \left(1 + \frac{\epsilon}{3}\right)\left(\frac{\epsilon}{16}\right) - 1 \\
&= 1 - \frac{\epsilon}{3} - \frac{\epsilon}{16} - \frac{\epsilon^2}{48} - 1 \\
&\geq 1 - \frac{\epsilon}{3} - \frac{\epsilon}{16} - \frac{\epsilon}{48} - 1 \\
&\geq 1 - \frac{5\epsilon}{12} - 1
\end{aligned}$$

So $\beta \geq 1 - \frac{5\epsilon}{12} > 1 - \epsilon$ which is the desired lower bound. Having shown the inequality 4.3 holds for $\mathbf{x} \in S^r_{\mathcal{B}}$, we have that the inequality holds for $\mathcal{L}^r_{\mathcal{B}}$ by re-scaling and reducing to the previous case. □

1 Since $\mathcal{L}^r_{\mathcal{B}} - \mathcal{L}^r_{\mathcal{B}} \subseteq \mathcal{L}^r_{\mathcal{B}}$, so by merit of Φ being a JL map of $\mathcal{L}^r_{\mathcal{B}}$ it is also a JL map of the set difference of $\mathcal{L}^r_{\mathcal{B}}$ with itself, and so Lemma 4.1.12 applies, and we have that Φ approximately preserves inner products of vectors in the subspace. So both norms and angles are preserved, and so in some sense the geometry of the subspace is preserved by the map.

2 Using the covering number bound in Lemma 4.2.7, we can conclude that $|C| \leq \left(\frac{3}{\frac{\epsilon}{16}}\right)^{2r} = \left(\frac{48}{\epsilon}\right)^{2r}$, where C is a minimal cover of the r dimensional unit sphere. Thus we can construct Φ where

$$m = \frac{c}{\epsilon^2} \log |C| \leq \frac{2cr}{\epsilon^2} \log \frac{48}{\epsilon}$$

Note we have optimal dependence on r. Note that the construction of Φ is oblivious - we do not need to know anything special about $\mathcal{L}^r_{\mathcal{B}}$; an upper bound on r will suffice in order to construct Φ

3 Because of 2, we have an $\epsilon - JL$ map with Rademacher entries if

$$m \geq \frac{c}{\epsilon^2} \log \left[\left(\frac{48}{\epsilon}\right)^{2r}\right] \geq \frac{c'r}{\epsilon^2} \log \left(\frac{48}{\epsilon}\right)$$

4 Fast $\sqrt{\frac{N}{m}}RFD$ ϵ-JL matrices have the row requirement

$$m = \frac{cr}{\epsilon^2}\log\left(\frac{48}{\epsilon}\right)\log^4 N$$

and the matrix vector multiplication time is $\mathcal{O}(N\log N)$.

Corollary 4.3.2. *Let $\epsilon \in (0,1)$. There exists an ϵ-JL map $\Phi \in \mathbb{C}^{m\times N}$ of any given r-dimensional subspace $\mathcal{L}_{\mathcal{B}} \subset \mathbb{C}^N$ with $m \leq \frac{Cr}{\epsilon^2}\log\left(1+\frac{32}{\epsilon}\right)$ where $C \in \mathbb{R}^+$ is an absolute constant (independent of all $r, \epsilon, m, N, \mathcal{L}_{\mathcal{B}}^r$)*

The proof using Corollary 4.2.8, Theorems 4.1.3 and 4.3.1 is left as an exercise.

Recall the following property of orthonormal matrices:

Suppose $B \in \mathbb{C}^{N\times r}$ is the matrix formed by writing the orthonormal basis elements of $\mathcal{L}_{\mathcal{B}}^r$ as columns:

$$B = \begin{pmatrix} | & | & & | \\ b_1 & b_2 & \dots & b_r \\ | & | & & | \end{pmatrix}$$

Then $\|B^*\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2$, $\forall \mathbf{x} \in \mathcal{L}_{\mathcal{B}}^r$.

In terms of the subject matter of this course, we can say that B is a 0-JL map. So why then are ϵ-JL maps of interest, if there is a common, well understood way to find lossless embeddings? In many settings however, we do not have detailed information about the subspace - for example we cannot easily find r linearly independent points, or in general sampling the space is costly.

Corollary 4.3.2 is an oblivious embedding; this means that we do not need to know what $\mathcal{L}_{\mathcal{B}}^r$ is in order to embed it into $\mathbb{C}^r$ accurately. One useful application that fits this setting is finding the (approximate) principle Eigenspace for huge matrices. Another application is a fast, approximate solution to least squares problems.

Exercise 4.3.1. *Prove Corollary 4.3.2.*

4.3.2 Overdetermined Least Squares

In the overdetermined least squares problem, we are tasked with finding a ℓ_2 minimizer $\mathbf{y}_{\min}$ to the matrix equation $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{C}^{N\times n}$, $N > n$, $\mathbf{b} \in \mathbb{C}^N$.

$$\mathbf{y}_{\min} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2$$

Define argmin? When $\mathbf{b}$ is in the range of A, then we will be able to find a solution which solves the equation exactly. However, if $\mathbf{b}$ does not lie in the range of A then $\mathbf{y}_{\min}$ will be the closest vector to $\mathbf{b}$ in the range of A.

A standard, classic solution approach to this uses QR decomposition and takes $\mathcal{O}(Nn^2)$ time. See Lecture 11 in [51]. We seek then then a solution approach which improves this runtime for $N \gg n$.

Theorem 4.3.3. *There exists universal constants $\bar{c}, c'$ such that a fast JL embedding matrix, $\sqrt{\frac{N}{m}}RFD \in \mathbb{C}^{m\times N}$ with*

$$m \geq \bar{c}(n+1)\ln\left(\frac{c'}{\epsilon}\right)\ln^4 N$$

will satisfy

$$(1-\epsilon)\|A\mathbf{y}-\mathbf{b}\|_2 \leq \sqrt{\frac{N}{m}}\|RFDA\mathbf{y}-RFD\mathbf{b}\|_2 \leq (1+\epsilon)\|A\mathbf{y}-\mathbf{b}\|_2$$

$\forall \mathbf{y} \in \mathbb{R}^n$ with probability at least $1-p-N^{-\ln^3 N}$

Explain better where lower bound on m is coming from.

Proof. Let $\mathcal{B} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{b}\}$ be the $n+1$ orthonormalized columns of A as well as $\mathbf{b}$. As before, let $\mathcal{L}_{\mathcal{B}}^{n+1}$ be the linear subspace spanned by the basis, and $S_{\mathcal{B}}^{n+1}$ the unit ball in the subspace. Let $C \subset S_{\mathcal{B}}^{n+1}$ be a minimal $\left(\frac{\epsilon}{16}\right)$ cover as in Theorem 4.3.1, and so $|C| \leq \left(\frac{48}{\epsilon}\right)^{n+1}$

Theorem 4.3.1 then implies that so long as Φ is a $\frac{\epsilon}{2}$-JL map of C then for each $\mathbf{y} \in \mathbb{C}^m$ we have $A\mathbf{y}-\mathbf{b} \in \text{span}(\mathcal{B})$ or equivalently $A\mathbf{y}-\mathbf{b} \in \mathcal{L}_{\mathcal{B}}^{n+1}$. So then for any such $\mathbf{y}$

$$(1-\epsilon)\|A\mathbf{y}-\mathbf{b}\|_2^2 \leq \sqrt{\frac{N}{m}}\|RFDA\mathbf{y}-RFD\mathbf{b}\|_2^2 \leq (1+\epsilon)\|A\mathbf{y}-\mathbf{b}\|_2^2$$

with probability at least $1-p-N^{-\ln^3 N}$ □

Denote $\tilde{F} = \sqrt{\frac{N}{m}}RFD$, the fast JL matrix from Theorem 4.3.3. How good is the approximation to the original least squares problem? Observe that $\tilde{F}A \in \mathbb{C}^{m\times n}$ and $\tilde{F}b \in \mathbb{C}^m$ we have then a compressed minimization problem,

$$\mathbf{y}'_{\min} = \arg\min_{\mathbf{z}\in\mathbb{C}^n} \|\tilde{F}A\mathbf{z}-\tilde{F}\mathbf{b}\|_2$$

By Theorem 4.3.3 we have

$$\begin{aligned}(1-\epsilon)\|A\mathbf{y}'_{\min}-\mathbf{b}\|_2^2 &\leq \|\tilde{F}A\mathbf{y}'_{\min}-\tilde{F}\mathbf{b}\|_2^2\\ &\leq \|\tilde{F}A\mathbf{y}_{\min}-\tilde{F}\mathbf{b}\|_2^2\\ &\leq (1+\epsilon)\|A\mathbf{y}_{\min}-\mathbf{b}\|_2^2\end{aligned}$$

Therefore,

$$\|A\mathbf{y}'_{\min}-\mathbf{b}\|_2 \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}\|A\mathbf{y}_{\min}-\mathbf{b}\|_2$$

Similarly we can bound from below,

$$\begin{aligned}(1+\epsilon)\|A\mathbf{y}'_{\min}-\mathbf{b}\|_2^2 &\geq \|\tilde{F}A\mathbf{y}'_{\min}-\tilde{F}\mathbf{b}\|_2^2\\ &\geq (1-\epsilon)\|A\mathbf{y}'_{\min}-\mathbf{b}\|_2^2\\ &\geq (1-\epsilon)\|A\mathbf{y}_{\min}-\mathbf{b}\|_2^2\end{aligned}$$

Thus the solution to the approximate answer has the following error bounds

$$\sqrt{\frac{1-\epsilon}{1+\epsilon}}\|A\mathbf{y}_{\min}-\mathbf{b}\|_2 \leq \|A\mathbf{y}'_{\min}-\mathbf{b}\|_2 \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}\|A\mathbf{y}_{\min}-\mathbf{b}\|_2.$$

This means that $\mathbf{y}'$ is almost as good a minimizer as the answer.

Runtime of the Randomized Approach:

The runtime of the approximate solution depends on multiplying A and b by $\tilde{F}$ and then also solving the compressed minimization problem. That is,

1. Computing $\tilde{F}A \in \mathbb{C}^{m\times n}$ can be computed in $\mathcal{O}\left(nN\log N\right)$ time
2. Computing $\tilde{F}\mathbf{b}$ can be computed in $\mathcal{O}\left(N\log N\right)$ time
3. Solving least squares problem $\arg\min_{\mathbf{z}\in\mathbb{C}^n}\|\tilde{F}A\mathbf{z}-\tilde{F}\mathbf{b}\|_2$ can be solved in $\mathcal{O}(mn^2)$ using QR factorization for example. Substituting in $m=\bar{c}(n+1)\ln\left(\frac{c'}{\epsilon}\right)\ln^4 N$ we have $\mathcal{O}(n^3\log^4 N)$ (constants depending on ϵ and p are collapsed)

So the total runtime to find $\mathbf{y}'_{\min}$ is $\mathcal{O}\left(nN\log N+n^3\log^4 N\right)$. Recall the classic solution requires $\mathcal{O}(n^2N)$. Therefore when $\log N \lesssim n \lesssim N\log^{-4} N$ we achieve a speedup by using the approximate solution approach. For example, should $n=\sqrt{N}$ then we have a runtime of $\mathcal{O}\left(N^{1.5}\log^4 N\right)$ for the approximate solution and $\mathcal{O}(N^2)$ for the classical solution approach.

4.3.3 A Sketching Method for Approximating Singular Values

Lemma 4.3.4. *Let $A\in\mathbb{C}^{N\times p}$ be a matrix with orthonormal columns. Suppose that $M\in\mathbb{C}^{m\times N}$ is an ϵ-JL map of the column space of $\{A\}$ into $\mathbb{R}^m$. Then MA is full rank with*

$$1-\epsilon\leq\sigma_j(MA)\leq 1+\epsilon \quad \forall j\in[p]. \tag{4.4}$$

Proof. Let $\mathbf{v}\in\mathbb{C}^p$ with $\|\mathbf{v}\|_2=1$ be such that $\sigma_j(MA)=\|MA\mathbf{v}\|_2$. Since

$$\begin{aligned}\|MA\mathbf{v}\|_2 &\leq (1+\epsilon)\|A\mathbf{v}\|_2 && (M \text{ is an } \epsilon\text{-JL map})\\ &= 1+\epsilon && (A \text{ has orthonormal columns}),\end{aligned}$$

we have $\sigma_j(MA)\leq 1+\epsilon$. Following the same reasoning we have that $\sigma_j(MA)\geq 1-\epsilon$. □

Lemma 4.3.5. *Let $V \in \mathbb{C}^{N\times r}$ have orthonormal columns, and suppose $M \in \mathbb{C}^{m\times N}$ is an $\epsilon-$JL map of $\mathcal{L}$ =Column span of V in $\mathbb{R}^N$. Then $MV \in \mathbb{C}^{m\times r}$ is full rank with*

$$1-\epsilon \leq \sigma_j(MV) \leq 1+\epsilon \qquad \forall j \in [r].$$

Proof. Let $u \in \mathbb{C}^r$ with $\|u\|_2 = 1$ be such that

$$\sigma_1(MV) = \|MVu\|_2.$$

u top singular vector of matrix... We have that:

$$\begin{aligned}\sigma_1(MV) = \|MVu\| &\leq (1+\epsilon)\|Vu\| \\ &= (1+\epsilon) \qquad \text{orthogonality of } V + \|u\|_2 = 1\end{aligned}$$

Similarly, letting $\mathbf{w} \in \mathbb{C}^r$ with $\|\mathbf{w}\|_2 = 1$ be such that

$$\sigma_r(MV) = \|MVw\|_2.$$

Then:

$$\begin{aligned}\sigma_r(MV) = \|MVw\| &\geq (1-\epsilon)\|Vw\| = (1-\epsilon). \\ &= (1-\epsilon) \qquad \text{orthogonality of } V + \text{normality of } u\end{aligned}$$

□

Theorem 4.3.6. *Suppose $A \in \mathbb{C}^{n\times p}$ is a rank $r \leq \min(n,p)$ matrix. Let $M \in \mathbb{C}^{m\times n}$ be an $\epsilon-$JL map of the column space of A into $\mathbb{C}^m$. Then:*

$$|\sigma_j(MA) - \sigma_j(A)| \leq \epsilon\sigma_j(A) \qquad \forall j \in [r]$$

Proof. Let $A = U\Sigma V^T$ be A's "thin" SVD. By Lemma 4.3.5:

$$\sigma_j(MU) \in (1-\epsilon, 1+\epsilon) \forall j \in [r].$$

Furthermore:

$$MA = MU\Sigma V^T$$

. By Theorem 2.3.9 we have:

$$\begin{aligned}\sigma_j(MA) &\leq \sigma_1(MU)\,\sigma_j\left(\Sigma V^T\right) \\ &\leq (1+\epsilon)\sigma_j(A) \quad \forall j \in [r]\end{aligned}$$

In addition, by Lemma 2.3.10 we have:

$$\begin{aligned}\sigma_j(MA) &\geq \sigma_r(MU)\,\sigma_j\left(\Sigma V^T\right) \\ &\geq (1-\epsilon)\sigma_j(A) \quad \forall j \in [r]\end{aligned}$$

□

Theorem 4.3.7. *Let $A \in \mathbb{C}^{N\times p}$ be a rank $r \leq \min(N,p)$ matrix. Suppose that $M \in \mathbb{C}^{m\times N}$ is an ϵ-JL map of the column space of A into $\mathbb{C}^m$. Then*

$$|\sigma_j(MA) - \sigma_j(A)| \leq \epsilon\sigma_j(A) \quad \forall j \in [r]. \tag{4.5}$$

Proof. Let $A = \underbrace{U}_{N\times r}\underbrace{\Sigma}_{r\times r}\underbrace{V^*}_{r\times p}$ be a thin SVD of A. By Lemma 4.3.4, the singular values of MU are contained in $[1-\epsilon, 1+\epsilon]$. Furthermore, by Lemma 2.3.10 we have that for all $j \in [r]$,

$$\sigma_j(MA) \leq \sigma_1(MU)\sigma_j(\Sigma V^*) \leq (1+\epsilon)\sigma_j(A) \tag{4.6}$$
$$\sigma_j(MA) \geq \sigma_r(MU)\sigma_j(A) \geq (1-\epsilon)\sigma_j(A). \tag{4.7}$$

The result now follows. □

Corollary 4.3.8. *Let $A \in \mathbb{C}^{N\times N}$ a rank r matrix. Suppose that M_1, M_2 is a ϵ-JL map of the column space of A, A^* into $\mathbb{C}^n$, respectively. Then*

$$|\sigma_j(M_1AM_2^*) - \sigma_j(A)| \leq \epsilon(2+\epsilon)\sigma_j(A) \quad \forall j \in [N]. \tag{4.8}$$

Proof. By Theorem 4.3.7 we have both

$$|\sigma_j(M_1AM_2^*) - \sigma_j(AM_2^*)| \leq \epsilon\sigma_j(AM_2^*)$$

and

$$|\sigma_j(AM_2^*) - \sigma_j(A)| = |\sigma_j(M_2A^*) - \sigma_j(A^*)| \leq \epsilon\sigma_j(A).$$

Therefore, by triangle inequality we obtain

$$\begin{aligned}|\sigma_j(M_1AM_2^*) - \sigma_j(A)| &\leq |\sigma_j(M_1AM_2^*) - \sigma_j(AM_2^*)| + |\sigma_j(AM_2^*) - \sigma_j(A)| \\ &\leq \epsilon(\sigma_j(AM_2^*) + \sigma_j(A)) \\ &\leq \epsilon((1+\epsilon)\sigma_j(A) + \sigma_j(A)) \\ &= \epsilon(2+\epsilon)\sigma_j(A).\end{aligned}$$

This completes the proof. □

Corollary 4.3.9. *Consider the following*

- *$A \in \mathbb{C}^{n\times n}$ be rank r.*
- *$M_1 \in \mathbb{C}^{m\times n}$ an $\epsilon - JL$ map of the column space of A into $\mathbb{C}^m$*
- *$M_2 \in \mathbb{C}^{m\times n}$ an $\epsilon - JL$ map of the column space of A^T into $\mathbb{C}^m$*

Then

$$\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A\right)\right| \leq \epsilon(2+\epsilon)\sigma_j\left(A\right) \qquad \forall j \in [r].$$

Proof. By Theorem 4.3.6, we will have

$$\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A M_2^T\right)\right| \leq \epsilon\sigma_j\left(A M_2^T\right) \quad \text{????since } col - space(AM_2^T) \subseteq col - space(A)$$

and

$$\begin{aligned}\left|\sigma_j\left(A M_2^T\right) - \sigma_j\left(A\right)\right| &= \left|\sigma_j\left(M_2 A^T\right) - \sigma_j\left(A^T\right)\right| \\ &\leq \epsilon\sigma_j\left(A^T\right) = \epsilon\sigma_j(A).\end{aligned}$$

Therefore

$$\begin{aligned}\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A\right)\right| &\leq \left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A M_2^T\right)\right| + \left|\sigma_j\left(A M_2^T\right) - \sigma_j\left(A\right)\right| \\ &\leq \epsilon\left(\sigma_j\left(A M_2^T\right) + \sigma_j\left(A\right)\right) \\ &\leq \epsilon\left((1+\epsilon)\sigma_j\left(A\right) + \sigma_j\left(A\right)\right) \\ &= \epsilon(2+\epsilon)\sigma_j\left(A\right).\end{aligned}$$

□

⇝ If $A \in \mathbb{C}^{n\times n}$ is rank $r \ll n$, we now get to compute the SVD of an $m \times m$ matrix (where m is $\tilde{\mathcal{O}}(r)$) to learn the approximate set of singular values of A.

⇝ This still works even if A is approximately low-rank.

Lemma 4.3.10. *Let $A \in \mathbb{C}^{n\times p}$ and suppose that $M \in \mathbb{C}^{m\times n}$ is an $\epsilon-$JL map of the columns of A;$\{a_j | j \in [p]\}$ into $\mathbb{C}^m$. Then*

$$\left|\|MA\|_F^2 - \|A\|_F^2\right| \leq \epsilon\|A\|_F^2.$$

Proof. Left as an exercise. add HW problem □

⇝ For an arbitrary matrix $A \in \mathbb{C}^{n\times n}$ we can always split A using its SVD

$$\begin{aligned}A = U\Sigma V^T &= U\left(\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & \Sigma_{n-r}\end{array}\right)V^T \\ &= U_r\Sigma_r V_r^T + U_{\setminus r}\Sigma_{n-r}V^T \\ &=: A_r + A_{\setminus r}.\end{aligned}$$

Here U_r and $U_{\setminus r}$ represent the matrix made of the fist r columns and last $n-r$ columns of U respectively. Also $U, \Sigma, V \in \mathbb{C}^{n\times n}$. actually define $A_{\setminus r}$

Theorem 4.3.11. *Let $A \in \mathbb{C}^{n\times n}$ and chose $r \in [n]$. Furthermore, suppose that $M_1 \in \mathbb{C}^{m\times n}$ and $M_2 \in \mathbb{C}^{m\times n}$ satisfy:*

1 M_1 is an $\epsilon-$JL map of the column space of A_r into $\mathbb{C}^m$

2 M_1 is an $\epsilon-$JL map of the n-columns of $A_{\setminus r}$ into $\mathbb{C}^m$

3 M_2 is an $\epsilon-$JL map of the column space of A_r^T into $\mathbb{C}^m$

4 M_2 is an $\epsilon-$JL map of the n-columns of of $A_{\setminus r}^T M_1^T$ into $\mathbb{C}^m$.

Then:

$$\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A\right)\right| \leq \epsilon(2+\epsilon)\sigma_j\left(A\right) + (1+\epsilon)\|A_{\setminus r}\|_F \qquad \forall j \in [r].$$

Proof.

$$M_1 A M_2 = M_1 A_r M_2^T + M_1 A_{\setminus r} M_2^T.$$

Thus by Theorem **??** (c) we have:

$$\begin{aligned}
\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(M_1 A_r M_2^T\right)\right| &\leq \sigma_1\left(M_1 A_r M_2^T\right) \\
&\leq \|M_1 A_{\setminus r} M_2^T\|_F = \|M_2 A_r^T M_1\|_F \\
&\text{Using Lemma 4.3.10 twice + together with assumption 2 and 4} \\
&\leq \sqrt{1+\epsilon}\|A_{\setminus r}^T M_1^T\|_F = \sqrt{1+\epsilon}\|M_1 A_{\setminus r}\|_F \\
&\leq (1+\epsilon)\|A_{\setminus r}\|_F
\end{aligned}$$

Finally we have:

$$\begin{aligned}
\left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(A\right)\right| &\leq \left|\sigma_j\left(M_1 A M_2^T\right) - \sigma_j\left(M_1 A_r M_2^T\right)\right| + \left|\sigma_j\left(M_1 A_r M_2^T\right) - \sigma_j\left(A\right)\right| \\
&\leq (1+\epsilon)\|A_{\setminus r}\|_F + \epsilon(2+\epsilon)\sigma_j(A) \qquad \text{Using Corollary 4.3.9.}
\end{aligned}$$

□

4.4 New Randomized SVD [CMSE 890 Lecture 14]

Definition 4.4.1. *Let $\epsilon > 0$ and $r \in \mathbb{N}$. A matrix $\tilde{X} \in \mathbb{C}^{n\times m}$ is an $(\epsilon, r)-$Projection Cost Preserving (PCP) sketch of $X \in \mathbb{C}^{n\times q}$ if for all orthogonal projection matrices $P \in \mathbb{C}^{n\times n}$ with rank at most r,*

$$(1-\epsilon)\|X - PX\|_F^2 \leq \|\tilde{X} - P\tilde{X}\|_F^2 \leq (1+\epsilon)\|X - PX\|_F^2$$

holds.

define orthogonal projection matrix in def above, and talk about it's structure as QQ^T

Algorithm 22 Compressed PCA

Input: $A \in \mathbb{C}^{n\times q}, \tilde{A} \in \mathbb{C}^{n\times m}$ and $(\epsilon, r) - PCP$ sketch of A.

Output: Approximation of A_r, the best rank r approximation of $A = U\Sigma V$ $(A_r = U_r\Sigma_r V_r)$.

Note: $\|A - A_r\|_F \leq \|A - V\|_F \quad \forall$ rank $\leq r$ matrices V.

1 Compute the SVD of $\tilde{A} = \underset{(n\times m)}{P}\underset{(m\times m)}{\Sigma}\underset{(m\times m)}{V}$ and let P_r be the first r columns of P

2 Compute the SVD of $\underset{(r\times q)}{P_r^* A} = \underset{(r\times r)}{\tilde{U}}\underset{(r\times r)}{\tilde{\Sigma}}\underset{(r\times q)}{\tilde{V}}$

return $\tilde{A}_r' = \underset{(n\times r)}{(P_r\tilde{U})}\underset{(r\times r)}{\tilde{\Sigma}}\underset{(r\times q)}{\tilde{V}}$ $\left(\text{will actually output } P_r\tilde{U}, \tilde{\Sigma}, \tilde{V}\right)$

$\rightsquigarrow$ A $QR-$based SVD can be computed in $\mathcal{O}(\min(n,q)\dot{n}q)-$time

$\rightsquigarrow$ The algorithm above will take $\mathcal{O}(\underset{line1}{m^2 n} + \underset{line2}{r^2 q} + \underset{compute P_r^* A}{rnq})-$time.

$\rightsquigarrow$ If $m \sim r$ sufficiently (it will) this gives us FIX!:

$$\mathcal{O}(n^3) - time \text{ (standard)} \qquad \text{V.S} \qquad \mathcal{O}(rnq + m^2 n + r^2 q) \text{ (algorithm above.)}$$

Theorem 4.4.2. *Let $\epsilon \in (0,1)$. Then the output of the compressed PCA Algorithm above will satisfy:*

$$\|A - \tilde{A}_r'\|_F \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}\|A - A_r\|_F.$$

Proof. Clean up presentation & make explanatory comments more clearly comments....

$$\begin{aligned}
\|A - \tilde{A}_r'\|_F^2 &= \|A - P_r\tilde{U}_r\tilde{\Sigma}\tilde{V}\|_F^2 = \|A - P_r P_r^* A\|_F^2 \\
&\leq \frac{\|\tilde{A} - P_r P_r^*\tilde{A}\|_F^2}{1-\epsilon} \quad \text{since } \tilde{A} \text{ is an } (\epsilon, r)\text{PCP sketch of } A \\
&\leq \frac{\|\tilde{A} - U_r U_r^*\tilde{A}\|_F^2}{1-\epsilon} \quad \text{by } P_r P_r^* \text{ def} + \underset{optimal rank r approximation}{\tilde{A}} = \tilde{A}_r \text{ where } A_r = U_r\Sigma_r V_r^T \\
&\leq \frac{1+\epsilon}{1-\epsilon}\|A - U_r U_r^* A\|_F^2 \qquad \text{using the PCP property} \\
&= \frac{1+\epsilon}{1-\epsilon}\|A - A_r\|_F^2.
\end{aligned}$$

□

$\rightsquigarrow$ Obvious question: How can we actually compute $\tilde{A}$ quickly ?

$\rightsquigarrow$ Idea:Let $\tilde{A} = \underset{(n\times N)}{A}\underset{(N\times m)}{\Omega^*}$ where Ω is a random matrix.

Make dimensions of matrices and variable names below consistent with definitions above...

Theorem 4.4.3. *Let $X \in \mathbb{C}^{n\times N}$ of rank $\tilde{r} \leq \min(n, N)$ have the full SVD $X = U\Sigma V^*$, and let $V_{r'} \in \mathbb{C}^{N\times r'}$ denote the first r' columns of $V \in \mathbb{C}^{N\times N}$ for all $r' \in [N]$. Fix $r \in [n]$ and consider the head-tail split $X = \underset{\text{best rk } r \text{ approx w.r.t} \|.\|_F}{X_r} + X_{\backslash r}$. If $\Omega \in \mathbb{C}^{m\times N}$ satisfies :*

1 $(1 - \frac{\epsilon}{3})\|X_r^* y\|_2^2 \leq \|\Omega X_r^* y\|_2^2 \leq (1 + \frac{\epsilon}{3})\|X_r^* y\|_2^2 \qquad \forall y \in \mathbb{R}^n.$

2 $\|X_{\backslash r}\Omega^*\Omega V_{\min(r,\tilde{r})} - X_{\backslash r}V_{\min(r,\tilde{r})}\|_F \leq \frac{\epsilon}{6\sqrt{\min(r,\tilde{r})}}\|X_{\backslash r}\|_F\|V_{\min(r,\tilde{r})}\|_F \leq \frac{\epsilon}{6}\|X_{\backslash r}\|_F,$

3 $(1 - \frac{\epsilon}{6})\|y\|_2^2 \leq \|\Omega y\|_2^2 \leq (1 + \frac{\epsilon}{6})\|y\|_2^2 \qquad \forall y \in \{n \text{ columns of } X_{\backslash r}^T\}$

4 $\|X_{\backslash r}\Omega^*\Omega X_{\backslash r}^* - X_{\backslash r}X_{\backslash r}^*\| \leq \frac{\epsilon}{6\sqrt{r}}\|X_{\backslash r}\|_F^2,$

Then $\tilde{X} = X\Omega^*$ is an $(\epsilon, r) - PCP$ sketch of X.

Proof. Let $Q \in \mathbb{C}^{n\times r}$ be an arbitrary matrix with orthogonal columns s.t QQ^* is an orthogonal projection matrix. Let $P = I - QQ^*$. It suffices to prove that :

(A) $\left|\|PX\|_F^2 - \|PX\Omega^*\|_F^2\right| \leq \epsilon\|PX\|_F^2$

Writing $X = X_r + X_{\backslash r}$ where ($X_r = \underset{\text{(first r cols of U in SVD of X)}(r\times r)}{U_r}\underset{(r\times N)}{\Sigma_r\ V_r^*}$) and using that

i $trAA^* = \|A\|_F^2$

ii $P = P^*$

iii $X_{\backslash r}X_r^* = 0$ (HW!)

We have that the LHS of (A) is equivalent to

$$\begin{aligned} LHS(A) =& \left|\|P(X_r + X_{\backslash r})\|_F^2 - \|P(X_r + X_{\backslash r})\Omega^*\|_F^2\right| \\ =& \left|tr\left(P(X_rX_r^* + X_{\backslash r}X_{\backslash r}^*)P\right) - tr(P(X_r\Omega^*\Omega X_r^* + X_r\Omega^*\Omega X_{\backslash r}^* + X_{\backslash r}\Omega^*\Omega X_r^* + X_{\backslash r}\Omega^*\Omega X_{\backslash r}^*)P^*)\right| \\ \leq& \left|tr\left(P(X_rX_r^* - X_r\Omega^*\Omega x_r^*)P\right)\right| + 2\left|tr\left(P(X_{\backslash r}\Omega^*\Omega X_r^*)P\right)\right| + \left|tr\left(P(X_{\backslash r}X_{\backslash r}^* - X_{\backslash r}\Omega^*\Omega X_{\backslash r}^*)P\right)\right| \end{aligned}$$

Thus it suffices to show that $(B) \leq \epsilon\|PX\|_F^2$. To show this it in turn suffices to show that

(I) $\left|tr(P(X_rX_r^* - X_r\Omega^*\Omega X_r^*)P)\right| = \left|\|X_r^*P\|_F^2 - \|\Omega X_r^*P\|_F^2\right| \leq \frac{\epsilon}{3}\|PX\|_F^2$(HW!)

(II) $\left|tr(P(X_{\setminus r}\Omega^*\Omega X_r^*)P)\right| \leq \frac{\epsilon}{6}\|PX\|_F^2$

(III) $\left|tr(P(X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*)P)\right| \leq \frac{\epsilon}{3}\|PX\|_F^2$

Proof of II: Using the invariance of trace under transposition and permutation and the the fact that $P^* = P = P^2$ we get

$$\begin{aligned}\left|tr(PX_{\setminus r}\Omega^*\Omega X_r^*P)\right| &= \left|tr(PX_r\Omega^*\Omega X_{\setminus r}^*P)\right| \\ &= \left|tr(PX_r\Omega^*\Omega X_{\setminus r}^*)\right|\end{aligned}$$

Using the thin SUD of $X = \underset{(n\times\tilde{r})}{\hat{U}}\underset{(\tilde{r}\times\tilde{r})}{\hat{\Sigma}}\underset{(\tilde{r}\times N)}{\hat{V}^*}$ we can write

$$\begin{aligned}&= \left|tr(P\hat{U}\hat{U}^*X_r\Omega^*\Omega X_{\setminus r}^*)\right| \\ &= \left|tr(P\hat{U}\hat{\Sigma}\hat{V}^*\hat{V}\hat{\Sigma}^{-1}\hat{U}^*X_r\Omega^*\Omega X_{\setminus r}^*)\right| \\ &= \left|tr(P\hat{U}\hat{\Sigma}\hat{V}^*)(\hat{V}\hat{\Sigma}^{-1}\hat{U}^*X_r\Omega^*\Omega X_{\setminus r}^*)\right| \\ &= \left|tr(PX)(\hat{V}\hat{\Sigma}^{-1}\hat{U}^*X_r\Omega^*\Omega X_{\setminus r}^*)\right|\end{aligned}$$

$\rightsquigarrow \langle A, B\rangle = tr(AB^*)$ is an inner product with $\|A\|_F = \sqrt{tr(AA^*)}$. Hence by C-S we have:

$$\begin{aligned}\left|tr(PX_{\setminus r}\Omega^*\Omega X_r^*P)\right| &\leq \|PX\|_F\|\hat{V}\hat{\Sigma}^{-1}\hat{U}^*X_r\Omega^*\Omega X_{\setminus r}^*\|_F \\ &= \|PX\|_F\|\hat{\Sigma}^{-1}\hat{U}^*(U_r\Sigma_rV_r^*)\Omega^*\Omega X_{\setminus r}^*\|_F \qquad \text{since } \hat{V} \text{ has orthogonal colu} \\ &= \|PX\|_F\|V_{\min(r,\tilde{r})}^*\Omega^*\Omega X_{\setminus}^*\|_F \\ &= \|PX\|_F\|X_{\setminus r}\Omega^*\Omega V_{\min(r,\tilde{r})}\|_F\end{aligned}$$

Now using that $X_{\setminus r}V_{\min(r,\tilde{r})} = 0$ we have

$$\begin{aligned}&= \|PX\|_F\|X_{\setminus r}\Omega^*\Omega V_{\min(r,\tilde{r})} - X_{\setminus r}V_{\min(r,\tilde{r})}\|_F \\ &\leq \|PX\|_F\frac{\epsilon}{6}\|X_{\setminus r}\|_F \qquad \text{by assumption (2)} \\ &\leq \frac{\epsilon}{6}\|PX\|_F^2\end{aligned}$$

Since $\|X_{\setminus r}\|_F = \|X - \underset{\text{best rk r approx}}{X_r}\|_F \leq \|X - QQ^*\|_F$.

Proof of III:

$$\begin{aligned}
\left|tr(P(X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*))\right| &= \left|tr((I-QQ^*)(X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*))\right| \\
&\leq \left|tr(X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*)\right| + \left|QQ^T(X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*)\right| \\
&\leq \left|\|X_{\setminus r}^*\|_F^2 - \|\Omega X_{\setminus r}^*\|_F^2\right| + \|QQ^T\|_F\|X_{\setminus r}X_{\setminus r}^* - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*\| \\
&\leq \frac{\epsilon}{6}\|X_{\setminus r}^*\|_F^2 + \sqrt{r}\|X_{\setminus r}X_{\setminus r}^T - X_{\setminus r}\Omega^*\Omega X_{\setminus r}^*\|_F \\
&\leq (\epsilon/6 + \epsilon/6)\|X_r\|_F^2 \\
&\leq \frac{\epsilon}{3}\|PX\|_F^2 \qquad \text{since } X_r \text{ is the best rank r approximation}
\end{aligned}$$

□

CMSE 890 Lecture 15

Lemma 4.4.4. *Let $A \in \mathbb{C}^{n\times N}$ have rank $\tilde{r} \leq \min(n,N)$. Fix $r \in [N]$. There exist finite sets $S_1, S_2 \subset \mathbb{C}^N$ (determined by A) with $|S_1| \leq \left(\frac{141}{\epsilon}\right)^{\min(r,\tilde{r})}$ and $|S_2| \leq 32n^2 + n$ such that the following holds: If $\Omega \in \mathbb{C}^{m\times N}$ is both:*

- *An $\epsilon/6-$JL map of S_1 into $\mathbb{C}^m$ and*
- *An $\frac{\epsilon}{6\sqrt{r}}-$JL map of S_2 into $\mathbb{C}^m$,*

then $A\Omega^*$ is an $(\epsilon, r)-$PCP sketch of A.

Proof. To ensure condition 1 of Theorem 4.4.3 Ω needs to be an oblivious subspace embedding of the column space of A_r^*. This can be accomplished by having Ω be an $\epsilon/6-$JL map of a minimal $(\epsilon/48)-$cover of the unit ℓ_2-ball in the $\min(r,\tilde{r})-$ dimensional column space of A_r^*. Let S_1 be this $(\epsilon/48)-$cover.
To guarantee condition 2 of Theorem 4.4.3 we need to embed the set

$$S' = \{x \pm iy, x \pm y | x, y \in S\}$$

where $S = \{\text{columns of} A_{\setminus r}^*\} \cup \{\text{columns of } V_{\min(r,\tilde{r})}\}$. Note that $|S| \leq 2n$ so $|S'| \leq 4(2n)^2 \leq 16n^2$.
Similarly to guarantee condition 4 of Theorem 4.4.3 we need to embed a second set $\tilde{S}'$ of size $\leq 16n^2$.
Finally to guarantee condition 3 we need to $\epsilon-$JL map $\tilde{\tilde{S}} = \{n\text{columsn of } A_{\setminus r}^*\}$ into $\mathbb{C}^m$.
Thus $S_2 := S' \cup \tilde{S}' \cup \tilde{\tilde{S}}$ has cardinality $\leq 32n^2 + n$. □

Example 4.4.5. *If* $\Omega \in \mathbb{C}^{m\times N}$ *has i.i,d* ± 1*(with prob 1/2) for entries then it will be an* $(\epsilon, r)-$*PCP sketch of a given* $A \in \mathbb{C}^{n\times N}$ *w.h.p if* $m \geq c\frac{r}{\epsilon^2}\max\{\ln\left(\frac{c_1}{\epsilon}\right), \ln(c_2 n)\}$ *where* $c, c_1, c_2 \in \mathbb{R}^+$ *are absolute constants.*

Make proving example and exercise... Add a few more special cases too!

Chapter 5

Sublinear-Time Compressive Sensing

BEGIN WITH GENRAL DISCUSSION OF COHERENCE AND SLOW COMPRESSIVE SENSING

5.1 "Slow" combinatorial compressive sensing using binary low coherence matrices

We will now discuss how coherence propertries can be combined with deterministic constructions to achieve compressive sensing with quadratic dependence on sparsity s (compared with linear dependence for probabilistic approaches seen so far) but which do offer faster recovery and also guarantee error bounds (i.e. no chance of failure)

Definition 5.1.1 (Coherence)**.** *Let $\Phi \in \mathbb{C}^{m \times N}$ be a matrix with ℓ^2-normalized columns $\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{N-1}$ having $\|\phi_j\| = 1 \ \forall j \in [N]$. The coherence $\mu(\Phi) = \max_{i \neq j} |\langle \boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i|$*

Note that if U is orthonormal, then $\mu(U) = 0$. If U contains two identical columns then $\mu(U) = 1$. So we see that $\mu(\Phi) \in [0, 1]$. Compared to other matrix properties of interest for compressive sensing settings, coherence is easy to compute.

We will show how low coherence matrices have the RIP property, and in turn can be used as JL maps. In order to do this, we will need the following classic theorem from linear algebra.

Theorem 5.1.2 (Gerschgorin Disc)**.** *Let λ be an eigenvalue of a square matrix $A \in \mathbb{C}^{N \times N}$. Then there exists an index $j \in [N]$ such that*

$$|\lambda - A_{jj}| \leq \sum_{\ell \in [N] \setminus \{j\}} |A_{j\ell}|$$

Proof. Let $(\lambda, \mathbf{u})$ be an eigenpair of the matrix. Let j be the index corresponding to the largest entry of the eigenvector, i.e. $|u_j| = \|\mathbf{u}\|_\infty$. Then $\sum_{\ell \in [N]} A_{j\ell} u_\ell = \lambda u_j$. That is since $A\mathbf{u} = \lambda \mathbf{u}$, the j-th entry of the vector $A\mathbf{u}$ is the inner product of the j-th row of A with $\mathbf{u}$ scaled by λ.

Now, moving the term $A_j j u_j$ to the other side, we obtain

$$\sum_{\ell \in [N] \setminus \{j\}} A_{j\ell} u_\ell = (\lambda - A_j j) u_j$$

Using the triangle inequality and bounding with the infinity norm, we have

$$|\lambda - A_{jj}||u_j| \leq \sum_{\ell \in [N] \setminus \{j\}} |A_{j\ell} u_\ell| \implies |\lambda - A_{jj}||u_j| \leq u_j \sum_{\ell \in [N] \setminus \{j\}} |A_{j\ell}|$$

Dividing each side then by $|u_j|$ yields the desired result. Note that every eigenpair may have a different center and radius, depending on which entry of the eigenvector is of largest magnitude. □

Corollary 5.1.3. *Every eigenvalue of A lies in at least one of the N circular disks in the complex plane with centers A_{jj} and radii $\sum_{i \neq j} |A_{ij}|$. Moreover if m of these disks form a connected domain that is disjoint from the other $N - m$ disks, then there are m eigenvalues of A within the domain.*

Theorem 5.1.4. *Let $\Phi \in \mathbb{C}^{m \times N}$ be a matrix with ℓ^2-normalized columns, take $s \in [N]$. Then $\forall s$-sparse vectors $\mathbf{x} \in \mathbb{C}^N$*

$$(1 - (s-1)\mu(\Phi))\|\mathbf{x}\|_2 \leq \|\Phi \mathbf{x}\|_2^2 \leq (1 + (s-1)\mu(\Phi))\|\mathbf{x}\|_2^2$$

Proof. Let $S \subset [N]$, $|S| \leq s$. Then the matrix $\Phi_S^* \Phi_S \in \mathbb{C}^{s \times s}$ formed by omitting all columns of Φ not in S is positive semi-definite. Denote its largest and smallest eigenvalues as $\lambda_{\max}, \lambda_{\min}$. If $\mathbf{x}$ is s-sparse and $S = \operatorname{supp}(\mathbf{x})$ then

$$\begin{aligned} \|\Phi \mathbf{x}\|_2^2 &= \|\Phi_S \mathbf{x}_S\|_2^2 \\ &= \langle \Phi_S \mathbf{x}_S, \Phi_S \mathbf{x}_S \rangle \\ &= \langle \Phi_S^* \Phi \mathbf{x}_S, \mathbf{x}_S \rangle \\ &\leq \lambda_{\max} \|\mathbf{x}_S\|_2^2 \\ &\leq \lambda_{\max} \|\mathbf{x}\|_2^2 \end{aligned}$$

In a similar fashion we can show that $\|\Phi \mathbf{x}\|_2^2 \geq \lambda_{\min} \|\mathbf{x}\|_2^2$. That is we have

$$\lambda_{\min} \|\mathbf{x}\|_2^2 \leq \|\Phi \mathbf{x}\|_2^2 \leq \lambda_{\max} \|\mathbf{x}\|_2^2$$

Gerschgorin's Disc theorem implies that there exists some index $j \in S$ such that

$$\left|\lambda - (\Phi_S^*\Phi_S)_{jj}\right| \leq \sum_{i\neq j}^{s} \left|(\Phi_S^*\Phi_S)_{ij}\right|$$

However, we know that $(\Phi_S^*\Phi_S)_{ij} = \langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j \rangle$, so

$$|\lambda - 1| \leq \sum_{i\neq j}^{s} |\langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j \rangle| \leq (s-1)\mu(\Phi)$$

□

Theorem 5.1.4 immediately implies the following

1. Φ has the RIP of order $(s, (s-1)\mu(\Phi))$.
2. For $\Phi_S^*\Phi_S \in \mathbb{C}^{N\times N}$, all of its non-zero eigenvalues are contained in the interval $[1-(s-1)\mu(\Phi_S^*\Phi_S), 1+(s-1)\mu(\Phi_S^*\Phi_S)]$

Definition 5.1.5. *Let $K, \alpha \in [N] := \{0, \ldots, N-1\}$. A matrix $A \in \{0,1\}^{m\times N}$ is* (K,α)-coherent *if the following conditions hold:*

1. *Every column of A contains at least K ones, and*
2. *For every $j, \ell \in [N]$, $j \neq \ell$, the inner product of the columns $\mathbf{a}_j$ and $\mathbf{a}_\ell$ satisfies $\langle \mathbf{a}_j, \mathbf{a}_\ell \rangle \leq \alpha$.*

Exercise 5.1.1. *Fix $\omega \in [N] \setminus \{0\}$ and let $X_\ell = \exp\left(\frac{2\pi i u_\ell \omega}{N}\right)$ where u_ℓ are i.i.d. uniformly in $[N]$ random variables $\forall \ell \in [m]$.*

1. *Prove that $0 = \mathbb{E}\left[\frac{1}{m}\mathrm{Re}(X_\ell)\right] = \mathbb{E}\left[\frac{1}{m}\mathrm{Im}(X_\ell)\right]$*
2. *Use Theorem 3.11.2 twice to show that*

$$P\left[\frac{1}{m}\left|\sum_{\ell=1}^{m} X_\ell\right| \geq t\right] \leq \frac{p}{N-1}$$

for any choice of $p \in (0,1)$ provided $m \geq \frac{4}{t^2}\ln\frac{4(n-1)}{p}$

3. *Let $A \in \mathbb{C}^{m\times N}$ be given by*

$$A_{\ell,\omega} = \frac{1}{\sqrt{m}}\exp\left(\frac{2\pi i u_\ell \omega}{N}\right),\ \ell \in [m],\ \omega \in [N]$$

Show that the columns of A are ℓ^2-normalized and that the coherence of $\mu(A) < \epsilon$ with probability greater than $1-p$ provided

$$m \geq \frac{4}{\epsilon^2}\ln\left(\frac{4(N-1)}{p}\right)$$

4. Show that A has the RIP of order (s, ϵ) for A with high probability when

$$m \geq C\frac{s^2}{\epsilon^2}\ln\left(\frac{4(N-1)}{p}\right)$$

Goal. *Keep α small while making K large.*

Proposition 5.1.6 (Welch bound)**.** *For a matrix $A \in \{0,1\}^{m\times N}$, the coherence satisfies*

$$\max_{1\leq j\neq \ell\leq N}\left|\left\langle \frac{\mathbf{a}_j}{\|\mathbf{a}_j\|_2}, \frac{\mathbf{a}_\ell}{\|\mathbf{a}_\ell\|_2}\right\rangle\right| \geq \sqrt{\frac{N-m}{m(N-1)}}.$$

The Welch bound then gives a lower bound on the number of rows for a (K, α)-coherent matrix:

$$\frac{\alpha}{K} \geq \sqrt{\frac{N-m}{m(N-1)}} \implies m \geq \frac{K^2}{\alpha^2}\frac{N-m}{N-1}.$$

When, for example, $m \leq N/2$ (which we henceforth assume), we must then have $m = \Omega(K^2/\alpha^2)$.

Example 5.1.7 ([28], Theorem 2)**.** *Fix some probability threshold $\sigma \in [0,1)$, and generate $M \in \{0,1\}^{m\times N}$ where each entry is i.i.d. Bernoulli, i.e.,*

$$m_{i,j} = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p, \end{cases}$$

where

$$p = \frac{\log_{4/\mathrm{e}}\left(\frac{3N^2}{1-\sigma}\right)}{K\,(1+o(1))}$$

for some $K \geq \alpha \geq 2\log_{4/\mathrm{e}}\left(\frac{3N^2}{1-\sigma}\right)$. Then M will be (K, α)-coherent with probability at least σ provided that $m \geq cK^2/\alpha$.

5.1.1 Deterministic block constructions

Example 5.1.8 ([18] and [33])**.**

1. *Choose a prime $p \in [N]$.*
2. *For each $j \in [N]$ we will consider the representation of j in base p, denoted*

$$j = j_0 + j_1 p + j_2 p^2 + \cdots + j_{\lceil \log_p N\rceil - 1} p^{\lceil \log_p N\rceil - 1},$$

where $j_0, \ldots, j_{\lceil \log_p N\rceil - 1} \in [p]$.

3. *Now, we map every* $j \in [N]$ *to the polynomial*

$$Q_j(x) := j_0 + j_1 x + \cdots + j_{\lceil \log_p N \rceil - 1} x^{\lceil \log_p N \rceil - 1},$$

over the finite field $\mathbb{Z}_p$.

4. *Define* $M = (\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) \in \{0,1\}^{p^2 \times N}$ *by*

$$m_{\ell + bp, j} = \begin{cases} 1 & \text{if } Q_j(b) = \ell, \\ 0 & \text{otherwise,} \end{cases}$$

of the form depicted in **??**.

We now consider the coherence of M,

$$\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle = |\{b \in [p] : Q_j(b) = Q_\ell(b) \iff (Q_j - Q_\ell)(b) = 0\}|.$$

Since $Q_j - Q_\ell$ *is a polynomial of degree at most* $\lceil \log_p N \rceil - 1$, *it can have at most* $\lceil \log_p N \rceil$ *zeros. Thus,* M *is a* $(p, \lceil \log_p N \rceil)$*-coherent matrix with* p^2 *rows separated into* p *blocks.*

Example 5.1.9 (Construction by error-correcting codewords)**.** *We can view the previous construction as a special case of the construction of a* (K, α)*-coherent matrix where we view the columns as binary error-correcting codewords with Hamming weight* K *by specifying a lower bound on the Hamming distance.*

Indeed, writing $M = (\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) \in \{0,1\}^{m \times N}$ *where each codeword* $\mathbf{m}_j \in \{0,1\}^m$ *has Hamming weight* K *(that is,* K *nonzero entries), we calculate* $\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle$ *in terms of the Hamming distance* $\Delta(\mathbf{m}_j, \mathbf{m}_\ell) := |\{i \in [m] : (\mathbf{m}_j)_i \neq (\mathbf{m}_\ell)_i\}|$. *Let* $i_k \in [m]$ *be an index of* $\mathbf{m}_j$ *such that* $(\mathbf{m}_j)_{i_k} = 1$ *The corresponding entry of* $\mathbf{m}_\ell$ *will either satisfy*

1. $(\mathbf{m}_\ell)_{i_k} = 1$, *and therefore this index increases* $\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle$ *by one or,*

2. $(\mathbf{m}_\ell)_{i_k} = 0$, *and therefore this index increases* $\Delta(\mathbf{m}_j, \mathbf{m}_\ell)$ *by one. Additionally, this "mismatched one" in* $\mathbf{m}_j$ *must have a corresponding "mismatched one" somewhere in* $\mathbf{m}_\ell$ *(since both codewords have the same Hamming weight) which again increases* $\Delta(\mathbf{m}_j, \mathbf{m}_\ell)$ *by one.*

Thus, after iterating through all K *ones in* $\mathbf{m}_j$, *we account for* $\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle$ *and exactly half of* $\Delta(\mathbf{m}_j, \mathbf{m}_\ell)$, *that is*

$$\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle = K - \frac{\Delta(\mathbf{m}_j, \mathbf{m}_\ell)}{2}.$$

A lower bound for the Hamming distance of $2(K - \alpha)$ *will then ensure that* M *is* (K, α)*-coherent.*

In 5.1.8, each column has a Hamming weight of exactly p *when viewed as an error-correcting codeword. The Hamming distance* $\Delta(\mathbf{m}_j, \mathbf{m}_\ell)$ *is twice the number of block indices*

b which are not zeros of $Q_j - Q_\ell$. By same argument as before, this "number of non-zeros" is at least by $p - \lceil \log_p N \rceil$, giving

$$\Delta(\mathbf{m}_j, \mathbf{m}_\ell) \geq 2\left(p - \lceil \log_p N \rceil\right).$$

Thus, in the context of error-correcting codewords, we have again shown that the matrix constructed in 5.1.8 is $\left(p, \lceil \log_p N \rceil\right)$-coherent.

Example 5.1.10 (A Fourier friendly construction [29, 30])**.** *Let*

$$p_0 = 1,\ p_1 = 2,\ p_2 = 3,\ p_3 = 5,\ \ldots,\ p_\ell = \text{the } \ell\text{th prime.}$$

1. *For some starting index $q \in \mathbb{N}$, fix the K sequential primes $p_q, \ldots, p_{q+K-1}$.*
2. *Define $M \in \{0,1\}^{\left(\sum_{\ell=0}^{K-1} p_{q+\ell}\right) \times N}$ with rows indexed by*

$$(\ell, h) \in ([q, q+K-1] \cap \mathbb{N}) \times [p_\ell]$$

by

$$m_{(\ell,h),j} = \begin{cases} 1 & \text{if } j \equiv h \bmod p_{q+\ell}, \\ 0 & \text{otherwise.} \end{cases}$$

?? *gives an example of this constructed matrix for starting index $q = 1$. We now obtain K blocks of rows (one for each prime) where each column contains exactly one 1 in each block.*

Considering the coherence of M, we calculate

$$\langle \mathbf{m}_j, \mathbf{m}_\ell \rangle = |\{p \in \{p_q, \ldots p_{q+K-1}\} : \ell \equiv j \bmod p\}| =: |R|.$$

By the Chinese Remainder Theorem, the product of all primes in R must divide $|\ell - j| < N$. This restricts R to sets of primes whose product is strictly less than N. Thus, for $j \neq \ell$,

$$\alpha := \min\{m \in [K] : p_q p_{q+1} \cdots p_{q+m} \geq N\}$$

provides an upper bound on the cardinality of R and therefore the coherence of M. Then,

$$p_q^\alpha \leq p_q p_{q+1} \cdots p_{q+\alpha-1} < N,$$

giving that $\alpha \leq \left\lfloor \log_{p_q} N \right\rfloor$, and therefore M is $\left(K, \left\lfloor \log_{p_q} N \right\rfloor\right)$-coherent. Additionally, if q is chosen so that $p_{q-1} < K \leq p_q$, bounds on q [1] and the prime number theorem give that $p_{K+q-1} = \mathcal{O}(K \log K)$, and therefore M has

$$m = \sum_{\ell=0}^{K-1} p_{q+\ell} = \mathcal{O}(K^2 \log K)$$

columns.

Theorem 5.1.11. *For all $\tilde{x} \in \mathbb{C}^N$ with $\|\tilde{x}\|_0 \leq s$, if we set $K = s \lfloor \log_s N \rfloor / \varepsilon$, then*

$$(1-\varepsilon)\|\tilde{x}\|_2^2 \leq \|M\tilde{x}\|_2^2 \leq (1+\varepsilon)\|\tilde{x}\|_2^2.$$

Proof. The coherence of $M/\sqrt{K}$ is bounded by ε/s which implies the restricted isometry property (RIP) of order s by standard arguments, e.g., [21, Theorem 5.3]. □

Now how is this matrix Fourier friendly? Let $\tilde{N} = \prod_{\ell=q}^{q+K-1} p_\ell$ and $\tilde{M} \in \{0,1\}^{m \times \tilde{N}}$ be as above with, e.g., $K \leq p_q < 2K$ (which is possible by Bertrand's postulate) where $K = s \lfloor \log_s N \rfloor / \varepsilon$. Recall that these assumptions imply $s \ll N \ll \tilde{N}$. Additionally let $\hat{\mathbf{f}} \in \mathbb{C}^{\tilde{N}}$ be such that $\sum_{j>N} |\hat{f}_j|$ is small (e.g., zero) and suppose that $\{\hat{f}_j\}_{j\in[N]}$ has a good s-sparse approximation (e.g., because it's s-sparse). In this case, we can use compressive sensing methods to recover $\hat{\mathbf{f}}$ using only the values of $M\hat{\mathbf{f}}$. Moreover, we can compute these values quickly by the following observations.

First, we see that

$$\tilde{M}\hat{\mathbf{f}} = \tilde{M}F_{\tilde{N}\times\tilde{N}}\left(F_{\tilde{N}\times\tilde{N}}^{-1}\hat{\mathbf{f}}\right) =: \tilde{M}F_{\tilde{N}\times\tilde{N}}\mathbf{f},$$

where $\mathbf{f} := \left\{f\left(\frac{2\pi j}{\tilde{N}}\right)\right\}_{j\in[\tilde{N}]}$ is the vector of $\tilde{N}$ equally spaced samples from $f(x) := \sum_{j\in[\tilde{N}]} \hat{f}_j \mathrm{e}^{\mathrm{i}jx}$. But note that

$$\tilde{M}F_{\tilde{N}\times\tilde{N}} = F_{\tilde{N}\times\tilde{N}}$$

and therefore each row in the product is the product of a discrete spike train with $F_{\tilde{N}\times\tilde{N}}$. For example, following the same indexing scheme as $\tilde{M}$,

$$\left(\tilde{M}F_{\tilde{N}\times\tilde{N}}\right)_{(\ell,h),k} = \frac{1}{\tilde{N}}\sum_{j=0}^{\frac{\tilde{N}}{p_{q+\ell}}-1} \mathrm{e}^{\frac{-2\pi\mathrm{i}(h+jp_{q+\ell})k}{\tilde{N}}} = \begin{cases} \frac{1}{p_{q+\ell}}\mathrm{e}^{\frac{-2\pi\mathrm{i}hk}{\tilde{N}}} & \text{if } k \equiv 0 \bmod \frac{\tilde{N}}{p_{q+\ell}}, \\ 0 & \text{otherwise.} \end{cases}$$

Note then that this product is extremely sparse, with each block corresponding to $p_{q+\ell}$ having at most $p_{q+\ell}$ nonzero columns, which makes for at most m nonzero columns. Additionally, the resulting structure of the product allows one to compute

$$\tilde{M}\hat{\mathbf{f}} = \tilde{M}F_{\tilde{N}\times\tilde{N}}\mathbf{f} = \begin{pmatrix} F_{p_q\times p_q}\left(f\left(\frac{2\pi j}{p_q}\right)\right)_{j\in[p_q]} \\ F_{p_{q+1}\times p_{q+1}}\left(f\left(\frac{2\pi j}{p_{q+1}}\right)\right)_{j\in[p_{q+1}]} \\ \vdots \\ F_{p_{q+K-1}\times p_{q+K-1}}\left(f\left(\frac{2\pi j}{p_{q+K-1}}\right)\right)_{j\in[p_{q+K-1}]} \end{pmatrix}$$

via K fast Fourier transforms of size at most $P_{q+K-1} = \mathcal{O}(K \log K)$. Thus, $\tilde{M}\hat{\mathbf{f}}$ can be computed in $\mathcal{O}(K^2 \log^2 K \log\log K)$ time if we have sampling access to f. By our assumption that $K = s \lfloor \log_s N \rfloor / \varepsilon$ for compressive sensing, we effectively compute $M\hat{\mathbf{f}}$ in

$$\mathcal{O}\left(\frac{s^2}{\log^2 s} \log^2 N \left(\log s + \log\log N\right)^2 \log\left(\log s + \log N\right)\right) = \mathcal{O}\left(s^2 \log^{2+\varepsilon} N\right) = \mathcal{O}\left(s^2 \log^3 N\right)$$

time, which is sublinear in N when $s \ll N$.

However, while we can compute the samples of $M\hat{\mathbf{f}}$ in sublinear time, standard compressive sensing algorithms are all $\mathcal{O}(N)$-time, so the entire recovery process will not be sublinear. The solution which we now pursue will be to avoid these standard RIP based recovery algorithms.

We will need to generalize the error bounds and analyses previously developed for general sublinear-time compressive sensing methods as part of the analysis of the recovery algorithms in [4]. These more general results will then let us recover good best s-term approximations to eigenvectors in sub-linear time.

Definition 5.1.12. *Let $K, \alpha \in [N] := \{0, \ldots, N-1\}$. A matrix $A \in \{0,1\}^{m \times N}$ is* (K, α)-coherent *if the following conditions hold:*

1. *Every column of A contains at least K ones, and*
2. *For every $j, \ell \in [N]$, $j \neq \ell$, the inner product of the columns $\mathbf{a}_j$ and $\mathbf{a}_\ell$ satisfies $\langle \mathbf{a}_j, \mathbf{a}_\ell \rangle \leq \alpha$.*

See Example 5.1.2 for a concrete illustration. Random constructions include, e.g., this one:

Running Example ([28], Theorem 2)**.** *A random matrix $M \in \{0,1\}^{m \times N}$ with i.i.d. Bernoulli random entries will be (K, α)-coherent with high probability under mild assumptions provided that $m \geq cK^2/\alpha$.*

There are also deterministic constructions too of course that are nearly as good, including those by, e.g., [33, 18, 29, 30]. More generally, one can easily prove the following lemma after recalling a couple basic definitions from the theory of error correcting codes.

Running Example. *Let $\mathbf{c}_j, \mathbf{c}_\ell \in \{0,1\}^m$. The Hamming weight of $\mathbf{c}_j$ is $\mathrm{wt}(\mathbf{c}_j) := \|\mathbf{c}_j\|_1$. Moreover, the Hamming distance between $\mathbf{c}_j$ and $\mathbf{c}_\ell$ is $\mathrm{d}(\mathbf{c}_j, \mathbf{c}_\ell) := \|\mathbf{c}_j - \mathbf{c}_\ell\|_1$. Any error correcting code $(\mathbf{c}_0, \ldots, \mathbf{c}_{N-1}) \in \{0,1\}^{m \times N}$ with constant Hamming weight $K = \mathrm{wt}(\mathbf{c}_j)$ $\forall j \in [N]$ and minimum Hamming distance $\Delta := \min_{j \neq \ell} \|\mathbf{c}_j - \mathbf{c}_\ell\|_1$ is also a $(K, K - \frac{\Delta}{2})$-coherent matrix. In fact, one can see that $\langle \mathbf{c}_j, \mathbf{c}_\ell \rangle = K - \frac{\|\mathbf{c}_j - \mathbf{c}_\ell\|_1}{2} \leq K - \frac{\Delta}{2}$ for all $j \neq \ell$.*

Given Example 5 one can see that there are in fact many deterministic constructions of (K, α)-coherent matrices waiting in the error correcting code literature. See, e.g., [37] for more related discussion.

5.1.2 Majority (s, K)-reconstructing matrices

We begin with some definitions and examples toward an alternative to the Restricted Isometry Property (RIP) often used in the compressive sensing and sparse approximation literature [21].

Running Example. *The matrix* $A \in \{0,1\}^{4\times 6}$ *given by*

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

is $(2,1)$*-coherent.*

Definition 5.1.13. *If* M *has at least* K *ones in every column, then* $M(n) \in \{0,1\}^{K\times N}$ *for* $n \in [N]$ *is the* $K \times N$ *submatrix of* M *created by selecting the first* K *rows of* M *with nonzero entries in the* n*th column.*

Definition 5.1.14. *If* M *has at least* K *ones in every column, then* $M'(n) \in \{0,1\}^{K\times(N-1)}$ *is the* $K \times (N-1)$ *submatrix of* $M(n)$ *created by removing its* n*th column.*

Running Example. *Suppose* $K = 2$*. Then, for* $n = 2$*,*

$$A = \left(\begin{array}{cccccc} 1 & 1 & \boxed{1} & 0 & 0 & 0 \\ 1 & 0 & \boxed{0} & 1 & 1 & 0 \\ 0 & 1 & \boxed{0} & 1 & 0 & 1 \\ 0 & 0 & \boxed{1} & 0 & 1 & 1 \end{array}\right) \mapsto A(2) = \left(\begin{array}{cccccc} 1 & 1 & \boxed{1} & 0 & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 & 1 & 1 \end{array}\right)$$

(column 2 is all ones)

and for $n = 5$*,*

$$A = \left(\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & \boxed{0} \\ 1 & 0 & 0 & 1 & 1 & \boxed{0} \\ 0 & 1 & 0 & 1 & 0 & \boxed{1} \\ 0 & 0 & 1 & 0 & 1 & \boxed{1} \end{array}\right) \mapsto A(5) = \left(\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & \boxed{1} \\ 0 & 0 & 1 & 0 & 1 & \boxed{1} \end{array}\right).$$

(column 5 is all ones)

Furthermore, for $n = 2$ *and* $n = 5$ *as above,*

$$A'(2) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad A'(5) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Definition 5.1.15. *For* $\mathbf{x} \in \mathbb{C}^N$*, after ordering its entries by magnitude*

$$|x_{j_1}| \geq |x_{j_2}| \geq \cdots \geq |x_{j_N}|$$

(where ties are broken for $|x_{j_i}| = |x_{j_{i+1}}|$ so that $j_i \leq j_{i+1}$), we define the index sets

$$S_{k,1} := \{j_1, \ldots, j_k\},\ S_{k,2} := \{j_{k+1}, \ldots, j_{2k}\},\ \ldots,\ S_{k,r} := \{j_{(r-1)k+1}, \ldots, j_N\},$$

for $r = \lfloor \frac{N-1}{k} \rfloor + 1$.

Definition 5.1.16. *For* $\mathbf{x} \in \mathbb{C}^N$ *and any index set* $S \subseteq [N]$ *we define* the restriction of $\mathbf{x}$ to S *denoted* $\mathbf{x}|_S \in \mathbb{C}^N$ *to be the vector with entries*

$$(\mathbf{x}|_S)_j := \begin{cases} x_j & \text{if } j \in S \\ 0 & \text{otherwise.} \end{cases}$$

Definition 5.1.17. *Given* $\mathbf{x} \in \mathbb{C}^N$*, the* s*-sparse vector* $\mathbf{x}_s \in \mathbb{C}^N$ *is defined as* $\mathbf{x}_s := \mathbf{x}|_{S_{s,1}}$.

We can now present our alternative to the RIP.

Definition 5.1.18 (Majority (s, K)-reconstructing)**.** *Let* $s \in \mathbb{N}$*, and* $M \in \{0,1\}^{m \times N}$ *have at least* $K \in [m]$ *ones in every column. We will say that* M *is* majority (s, K)-reconstructing for $\mathbf{x} \in \mathbb{C}^N$ *if the set*

$$B_n := \left\{ j : \left| (M(n)\mathbf{x})_j - x_n \right| \leq (1/s) \left\| \mathbf{x} - \mathbf{x}_s \right\|_1 \right\} \subseteq [K] \tag{5.1}$$

has cardinality $|B_n| > K/2$ *for all* $n \in [N]$.

More generally, one can change the fraction involved in the cardinality lower-bound of B_n in Definition 5.1.18 from $K/2$ to $\frac{c-2}{c}K$ for any $c \geq 4$. Doing so allows for modified reconstruction procedures in the next section. Finally, the following lemma concerning majority (s, K)-reconstructing matrices will be useful later.

Lemma 5.1.19. *Suppose* $M \in \{0,1\}^{m \times N}$ *is majority* (s, K)*-reconstructing for* $\mathbf{x} \in [0, \infty)^N$*. Choose any diagonal matrix* $D \in \mathbb{C}^{N \times N}$ *you like satisfying* $|D_{j,j}| = 1$ *for all* $j \in [N]$*. Then,* M *is also* (s, K)*-reconstructing for* $D\mathbf{x} \in \mathbb{C}^N$.

Proof. Let $n \in [N]$, and choose any j such that $\left| \big(M(n)\mathbf{x}\big)_j - x_n \right| \leq \frac{1}{s} \|\mathbf{x} - \mathbf{x}_s\|_1$. Then,

$$\begin{aligned} \left| \big(M(n)D\mathbf{x}\big)_j - \big(D\mathbf{x}\big)_n \right| &= \left| \sum_{\ell \neq n} M_{j,\ell} \big(D\mathbf{x}\big)_\ell \right| \leq \sum_{\ell \neq n} \left| M_{j,\ell} \big(D\mathbf{x}\big)_\ell \right| = \sum_{\ell \neq n} M_{j,\ell} |x_\ell| \\ &= \left| \big(M(n)\mathbf{x}\big)_j - x_n \right| \leq \frac{1}{s} \|\mathbf{x} - \mathbf{x}_s\|_1 = \frac{1}{s} \left\| \big(D\mathbf{x}\big) - \big(D\mathbf{x}\big)_s \right\|_1. \end{aligned}$$

□

Algorithm 23 Median Recovery, MR : $\mathbb{C}^m \times \mathcal{P}([N]) \times [N] \to \mathbb{C}^N$.

INPUT: $\mathbf{y} = M\mathbf{x} + \mathbf{n}$, $S \subseteq [N]$, $s \in [N]$.
OUTPUT: $\mathbf{z}|_{\tilde{S}} \in (\mathbb{C} \times [N])^{\min(|S|,2s)}$, encoding a sparse vector $\mathbf{z} \in \mathbb{C}^N$

1: **for** $n \in S$ **do**
2: Let $\mathrm{Re}(z_n) \leftarrow$ median of $\mathrm{Re}\left(M(n)\mathbf{x} + \mathbf{n}|_{M(n)}\right)$ entries in $\mathbf{y}$.
3: Let $\mathrm{Im}(z_n) \leftarrow$ median of $\mathrm{Im}\left(M(n)\mathbf{x} + \mathbf{n}|_{M(n)}\right)$ entries in $\mathbf{y}$.
4: **end for**
5: Sort $\{z_n\}_{n\in S}$ by magnitude so that $|z_{n_1}| \geq \ldots \geq |z_{n_{|S|}}|$. $\tilde{S} := \{n_1, \ldots, n_{\min(2s,|S|)}\}$.
6: Output $\mathbf{z}|_{\tilde{S}}$ with

$$\left(\mathbf{z}|_{\tilde{S}}\right)_j := \begin{cases} z_j & \text{if } j \in \tilde{S} \\ 0 & \text{otherwise} \end{cases}$$

as a sparse approximation to $\mathbf{x}$.

5.1.3 A Reconstruction Algorithm

For measurements of a vector taken with a s-reconstructing matrix, we provide Algorithm 23 to rapidly construct an approximation. In the algorithm, $\mathbf{n} \in \mathbb{C}^m$ represents arbitrary additive errors on our measurements of $\mathbf{x}$ given by $\mathbf{y} = M\mathbf{x} + \mathbf{n}$, and $\mathbf{n}|_{M(n)} \in \mathbb{C}^K$ contains the K entries of $\mathbf{n} \in \mathbb{C}^m$ associated with the K rows of $M(n)$ in M.

Complexity Analysis

The runtime of Algorithm 23 can be accounted for as follows:

- The "for loop" lines 1–4 can be performed in $\mathcal{O}(|S| \cdot K)$-time using fast median algorithms [2] assuming that the rows of $M(n)$ can be identified in $\mathcal{O}(K)$-time for any given $n \in [N]$. Note that this is indeed the case for, e.g., the Fourier-friendly matrices in [29, 30] by simply computing n modulo K prime numbers.
- Line 5 can be performed in $\mathcal{O}(|S| \log |S|)$ time using, e.g., merge sort [36].
- Lines 5–6 output $\mathbf{z}|_{\tilde{S}}$ in a compressed format in $\mathcal{O}(s)$-time and space.

Therefore, the total runtime/flop count of Algorithm 23 is $\mathcal{O}\left(m + |S| \cdot \max\left(K, \log |S|\right)\right)$. Thus, the algorithm is fast if both $|S|$ and K are small. In Section 5.1.3, we will analyze when it is also accurate. And, in Section 5.1.6 we will discuss how to quickly identify a small S that still guarantees accuracy. Finally, it's also worth mentioning that Algorithm 23 is trivially parallelizable since the z_n values can be computed in parallel, and since efficient parallel sorting methods exist (see, e.g., [43] for a comparison of several standard parallelized sorting algorithms).

Theoretical Error Analysis

The following theorem provides approximation guarantees for the output of Algorithm 23.

Theorem 5.1.20 (Modified from [4])**.** *Let $\beta \in [1, \infty)$, $s \in [N]$, $\mathbf{x} \in \mathbb{C}^N$, and $\mathbf{n} \in \mathbb{C}^m$. Suppose that $M \in \{0,1\}^{m \times N}$ is majority (s, K)-reconstructing for $\mathbf{x} \in \mathbb{C}^N$, and that $S \subseteq [N]$ contains the set*

$$\mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n}) := \left\{ n \in [N] : |x_n| > \beta \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right) \right\}.$$

Then,

$$\|\mathbf{x} - \mathrm{MR}(M\mathbf{x} + \mathbf{n}, S, s)\|_2 \leq \|\mathbf{x} - \mathbf{x}_{2s}\|_2 + C_\beta \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{\sqrt{s}} + \sqrt{s}\|\mathbf{n}\|_\infty \right)$$

for an absolute constant $C_\beta \in \mathbb{R}^+$ depending only on β. Furthermore, $C_\beta \leq 6 + 2\sqrt{2}\beta$.

We will prove Theorem 5.1.20 with the help of a couple of lemmas.

Lemma 5.1.21. *Every z_n estimate produced in lines 1–4 of Algorithm 23 satisfies*

$$|z_n - x_n| \leq \sqrt{2} \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right).$$

Proof. This follows directly from the majority (s, K)-reconstructing property of M for $\mathbf{x} \in \mathbb{C}^N$. Indeed, since the set B_n in (5.1) of Definition 5.1.18 has cardinality $|B_n| > K/2$, reordering the K-length vector $\mathrm{Re}(M(n)\mathbf{x} + \mathbf{n}|_{M(n)})_{j \in [K]}$ by magnitude ensures that there exist indices in B_n of elements in the reordered vector which lay on either side of the median $\mathrm{Re}(z_n)$. Thus, there exists some $j \in B_n$ such that

$$\begin{aligned} |\mathrm{Re}(z_n) - \mathrm{Re}(x_n)| &\leq \left|\mathrm{Re}(M(n)\mathbf{x} + \mathbf{n}_{M(n)})_j - \mathrm{Re}(x_n)\right| \leq |(M(n)\mathbf{x})_j + n_j - x_n| \\ &\leq \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty =: \delta' \end{aligned} \tag{5.2}$$

where the first inequality holds by the previous argument, and the third inequality holds by the definition of B_n. Similarly, $|\mathrm{Im}(z_n) - \mathrm{Im}(x_n)| \leq \delta'$. Thus,

$$|z_n - x_n| \leq \sqrt{(\delta')^2 + (\delta')^2} = \sqrt{2}\delta'. \qquad \square$$

Lemma 5.1.22. *If $n \in \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n}) \setminus \tilde{S}$ for $\tilde{S}$ in line 5 of Algorithm 23, then*

$$|x_n| \leq \left(\beta + 2\sqrt{2}\right) \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right).$$

Proof. Note first that under the reordering $|x_{j_1}| \geq \ldots \geq |x_{j_N}|$, $j_\ell \in \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})$ implies that $j_k \in \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})$ for all $1 \leq k \leq \ell$. When $N > 2s$,

$$\|\mathbf{x} - \mathbf{x}_s\|_1 \geq \sum_{\ell=s+1}^{2s} |x_{j_\ell}| \geq s|x_{j_{2s+1}}|.$$

Thus, $\mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n}) \subseteq S_{2s,1} \cap S$ for all $\beta \geq 1$, giving $|\mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})| \leq \min(2s, |S|) = |\tilde{S}|$. Therefore if $n \in \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n}) \setminus \tilde{S}$, there must exist some $j \in \tilde{S} \setminus \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})$ in particular satisfying

$$|x_j| \leq \beta \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right) \quad \text{and} \quad |z_j| \geq |z_n|.$$

Hence, for δ' as in (5.2),

$$\beta\delta' + \sqrt{2}\delta' \geq |x_j| + \sqrt{2}\delta' \geq |z_j| \geq |z_n| \geq |x_n| - \sqrt{2}\delta'$$

where the second and fourth inequalities hold by Lemma 5.1.21. □

Lemma 5.1.23. *The distance between* $\mathbf{x}$ *and* $\mathbf{x}|_{\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})}$ *in* ℓ_2 *can be bounded by*

$$\left\| \mathbf{x} - \mathbf{x}|_{\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})} \right\|_2 \leq \|\mathbf{x} - \mathbf{x}_{2s}\|_2 + \sqrt{2s}\beta \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right).$$

Proof. Since $\mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n}) \subseteq S_{2s,1}$,

$$\left\| \mathbf{x} - \mathbf{x}|_{\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})} \right\|_2^2 = \|\mathbf{x} - \mathbf{x}_{2s}\|_2^2 + \sum_{n \in S_{2s,1} \setminus \mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})} |x_n|^2 \leq \|\mathbf{x} - \mathbf{x}_{2s}\|_2^2 + 2s\beta^2 \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right)^2.$$

□

We are now prepared to prove our theorem concerning the output of the Median Recovery algorithm.

Proof of Theorem 5.1.20. Let $\delta' := \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty$ as in (5.2). Then,

$$\begin{aligned} \|\mathbf{x} - \mathrm{MR}(M\mathbf{x} + \mathbf{n}, S, s)\|_2 &\leq \left\| \mathbf{x} - \mathbf{x}|_{\tilde{S}} \right\|_2 + \left\| \mathbf{x}|_{\tilde{S}} - \mathbf{z}|_{\tilde{S}} \right\|_2 \\ &\leq \left\| \mathbf{x} - \mathbf{x}|_{\tilde{S}} \right\|_2 + \sqrt{2s}(\sqrt{2}\delta'), \end{aligned}$$

by Lemma 5.1.21. Continuing with our upper bound using that $\tilde{S}^c = (\tilde{S}^c \setminus \mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})) \cup (\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n}) \setminus \tilde{S}) \subseteq (\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n}))^c \cup (\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n}) \setminus \tilde{S})$ we next obtain that

$$\begin{aligned}
\|\mathbf{x} - \mathrm{MR}(M\mathbf{x}+\mathbf{n}, S, s)\|_2 &\leq \left\|\mathbf{x} - \mathbf{x}|_{\tilde{S}}\right\|_2 + 2\sqrt{s}\delta' \;=\; \sqrt{\sum_{n\in\tilde{S}^c} |x_n|^2} + 2\sqrt{s}\delta' \\
&\leq \sqrt{\sum_{n\in\tilde{S}^c\setminus\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})} |x_n|^2} + \sqrt{\sum_{n\in\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})\setminus\tilde{S}} |x_n|^2} + 2\sqrt{s}\delta' \\
&\leq \sqrt{\sum_{n\in(\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n}))^c} |x_n|^2} + \sqrt{\sum_{n\in\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})\setminus\tilde{S}} |x_n|^2} + 2\sqrt{s}\delta' \\
&= \left\|\mathbf{x} - \mathbf{x}|_{\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})}\right\|_2 + \sqrt{\sum_{n\in\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})\setminus\tilde{S}} |x_n|^2} + 2\sqrt{s}\delta'.
\end{aligned}$$

Continuing our bound by applying Lemmas 5.1.23 and 5.1.22 we can now finally see that

$$\begin{aligned}
\|\mathbf{x} - \mathrm{MR}(M\mathbf{x}+\mathbf{n}, S, s)\|_2 &\leq \left\|\mathbf{x} - \mathbf{x}|_{\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})}\right\|_2 + \sqrt{\sum_{n\in\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})\setminus\tilde{S}} |x_n|^2} + 2\sqrt{s}\delta' \\
&\leq \|\mathbf{x} - \mathbf{x}_{2s}\| + \sqrt{\sum_{n\in\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})\setminus\tilde{S}} |x_n|^2} + (2+\sqrt{2}\beta)\sqrt{s}\delta' \\
&\leq \|\mathbf{x} - \mathbf{x}_{2s}\| + \sqrt{2s\left((\beta+2\sqrt{2})\delta'\right)^2} + (2+\sqrt{2}\beta)\sqrt{s}\delta' \\
&= \|\mathbf{x} - \mathbf{x}_{2s}\|_2 + \left(\sqrt{2}(\beta+2\sqrt{2}) + 2 + \sqrt{2}\beta\right)\sqrt{s}\delta'
\end{aligned}$$

as desired. □

Corollary 5.1.24. *Under the assumptions of Theorem 5.1.20, we also have*

$$\|\mathbf{x} - \mathrm{MR}(M\mathbf{x}, S, s)\|_2 \leq C'_\beta \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{\sqrt{s}}$$

when $\mathbf{n} = 0$. *Here,* $C'_\beta \in \mathbb{R}^+$ *is an absolute constant with* $C'_\beta \leq C_\beta + 1$ *for* C_β *the constant in Theorem 5.1.20.*

Proof. By [21, Proposition 2.3] (with $q = 2$ and $p = 1$), we have

$$\|\mathbf{x} - \mathbf{x}_{2s}\|_2 \leq \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{\sqrt{s}},$$

finishing the proof. □

Up to now, we have seen that there are some nice (K, α)-coherent matrix constructions, and that the majority (s, K)-reconstructing property allows for standard compressive sensing error guarantees to be obtained. We will now

1. relate (K, α)-coherent matrices to the majority (s, K)-reconstructing property by showing how to construct matrices with this property using any (K, α)-coherent matrix, and then
2. consider fast algorithms for *rapidly* finding small sets $S \supseteq \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})$. This will allow the Median Recovery algorithm to run in sublinear-time while still having good error bounds.

5.1.4 Constructions of majority (s, K)-reconstructing matrices

We will give a majority (s, K)-reconstructing matrix construction using (K, α)-coherent matrices. It will hold for all $\mathbf{x} \in \mathbb{C}^N$ deterministically, but will have a sub-optimal number of rows. This will suffice for our purposes herein. We note however that majority (s, K)-reconstructing matrices for *a fixed individual* $\mathbf{x} \in \mathbb{C}^N$ having fewer rows also exist (see, e.g., [28, Corollary 2] for an implicit construction).

We begin with the following lemmas to help with the deterministic construction.

Lemma 5.1.25. *Suppose $M \in \{0,1\}^{m \times N}$ is (K, α)-coherent. Let $n \in [N]$, $s \in [1, K/\alpha] \cap \mathbb{N}$, and $\mathbf{x} \in \mathbb{C}^{N-1}$. Then at most $s\alpha$ of the entries in $M'(n)\mathbf{x} \in \mathbb{C}^K$ will have magnitude greater than or equal to $\|\mathbf{x}\|_1/s$.*

Proof. By Markov's inequality, we have

$$\begin{aligned}\left|\left\{j : |(M'(n)\mathbf{x})_j| \geq \frac{\|\mathbf{x}\|_1}{s}\right\}\right| &\leq \frac{s}{\|\mathbf{x}\|_1}\left\|M'(n)\mathbf{x}\right\|_1 \\ &\leq s\left\|M'(n)\right\|_{1\to 1}.\end{aligned}$$

Furthermore, if we write $M'(n) = (\mathbf{m}'_k)_{k\in[N-1]}$ and $M(n) = (\mathbf{m}_\ell)_{\ell\in[N]}$ both column-wise, we may calculate

$$\begin{aligned}\left\|M'(n)\right\|_{1\to 1} &= \max_{k\in[N-1]}\left\|\mathbf{m}'_k\right\|_1 \\ &= \max_{\ell\in[N]\setminus\{n\}}\langle \mathbf{m}_\ell, \mathbf{m}_n\rangle \\ &\leq \alpha,\end{aligned}$$

finishing the proof. □

Lemma 5.1.26. *Suppose $M \in \{0,1\}^{m\times N}$ is a (K, α)-coherent matrix. Let $n \in [N]$, $s \in [1, K/\alpha] \cap \mathbb{N}$, $S \subset [N]$ with $|S| \leq s$, and $\mathbf{x} \in \mathbb{C}^{N-1}$. Then $M'(n)\mathbf{x}$ and $M'(n)\left(\mathbf{x} - \mathbf{x}|_S\right)$ will differ in at most $s\alpha$ of their K entries.*

Proof. Let $\mathbb{1} \in \mathbb{C}^{N-1}$ be the vector of all ones. We have for $\mathcal{B} := \{j : (M'(n)\mathbf{x})_j \neq (M'(n)(\mathbf{x} - \mathbf{x}|_S))_j\}$ that

$$|\mathcal{B}| = \left|\left\{j : \left(M'(n)\mathbf{x}|_S\right)_j \neq 0\right\}\right| \leq \left|\left\{j : \left(M'(n)\mathbb{1}|_S\right)_j \geq 1\right\}\right| \tag{5.3}$$

since the nonzero entries of $M'(n)$ are all ones. Now, we may apply Lemma 5.1.25 to (5.3) with $M'(n)$ applied to $\mathbb{1}|_S$, which has $\|\mathbb{1}|_S\|_1 = |S| \leq s$ to learn that $|\mathcal{B}| \leq \alpha s$. □

We are now prepared to provide our general majority (s, K)-reconstructing matrix construction.

Theorem 5.1.27 (Modified from [4])**.** *Suppose M is (K, α)-coherent. Let $s \in [1, K/\alpha] \cap \mathbb{N}$ and $c \in [4, \infty) \cap \mathbb{N}$. If $K > c\alpha s$, then M is majority (s, K)-reconstructing for all $\mathbf{x} \in \mathbb{C}^N$. In particular, the cardinality of B_n in* (5.1) *is such that $|B_n| > \left(\frac{c-2}{c}\right) K$ for all $n \in [N]$ and $\mathbf{x} \in \mathbb{C}^N$.*

Proof. Let $n \in [N]$ and $\mathbf{x} \in \mathbb{C}^N$. Furthermore, let $\mathbf{y} \in \mathbb{C}^{N-1}$ be $\mathbf{x}$ with x_n removed so that

$$y_j = \begin{cases} x_j & \text{if } j < n \\ x_{j+1} & \text{if } j \geq n. \end{cases}$$

Finally, let $\mathbf{m}_0, \ldots \mathbf{m}_{N-1} \in \{0, 1\}^K$ be the columns of $M(n)$.

We have that

$$M(n)\mathbf{x} = x_n \mathbf{m}_n + M'(n)\mathbf{y} = x_n \mathbb{1} + M'(n)\mathbf{y}.$$

Lemma 5.1.26 tells us that at most $s\alpha$ entries of $M'(n)\mathbf{y}$ differ from those in $M'(n)\left(\mathbf{y} - \mathbf{y}_s\right)$. Of the remaining entries of $M'(n)\mathbf{y}$ (of which there are at least $K - s\alpha$), at most $s\alpha$ of them have magnitudes greater than or equal to $\|\mathbf{y} - \mathbf{y}_s\|_1 / s$ by Lemma 5.1.25 (since removing the at most $s\alpha$ rows j from M for which $\left(M'(n)\left(\mathbf{y} - \mathbf{y}_s\right)\right)_j \neq \left(M'(n)\mathbf{y}\right)_j$ leaves us with another $(K - s\alpha, \alpha)$-coherent matrix and $s \in [1, \frac{K - s\alpha}{\alpha}]$ as $K > c\alpha s > 2\alpha s$). Hence, at least

$$K - 2s\alpha = K - \frac{2}{c} c\alpha s > K - \frac{2}{c} K = \left(\frac{c-2}{c}\right) K$$

entries of $M'(n)\mathbf{y}$ have magnitudes bounded above by

$$\frac{1}{s}\|\mathbf{y} - \mathbf{y}_s\|_2 \leq \frac{1}{s}\|\mathbf{x} - \mathbf{x}_s\|_1.$$

The result follows after noting that $\frac{c-2}{c} \geq \frac{1}{2}$ for all $c \in [4, \infty)$. □

Recalling the Fourier-friendly matrices from [29, 30], we note that setting $K = 4s \lfloor \log_s N \rfloor$ for them yields a majority (s, K)-reconstructing matrix for any $s \leq N$. This matrix has at most

$$m = \mathcal{O}\left(s^2 \log_s^2 N \log(s \log_s N)\right)$$

rows.

5.1.5 Toward Fast Support Recovery: Generalized Bit Testing

Our current objective is to rapidly identify a small superset S of the set $\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})$ defined in Theorem 5.1.20. In what follows we will learn an encoded version of each element in $\mathcal{C}_{s,\beta}(\mathbf{x},\mathbf{n})$ using Error Correcting Code (ECC) ideas. To begin, note that any (K,α)-coherent matrix $A \in \{0,1\}^{m\times N}$ with $\alpha < K$ can't have any repeated columns since $\langle \mathbf{a}_j, \mathbf{a}_\ell\rangle = K > \alpha$ if $\mathbf{a}_j = \mathbf{a}_\ell$. Thus, for any such matrix the function $f_A : [N] \to \{0,1\}^m$ defined by $f(j) = A_{:,j}$ (i.e., that maps column numbers to the columns of A) is injective. In what follows below, we also want $f_A^{-1} : \{\text{columns of } A\} \subset \{0,1\}^m \to [N]$ to be rapidly computable for such matrices. An example of a class of matrices with this property follows.

Definition 5.1.28. *For $N \in \mathbb{N}$ the N^{th} bit testing matrix, $B_N \in \{0,1\}^{(1+\lceil \log_2 N\rceil)\times N}$, is the matrix whose j^{th}-column $\forall j \in [N]$ is a 1 followed by j written in binary. For example,*

$$B_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Recalling that we begin indexing the rows of our matrices with 0, we can see now that

$$j = f_{B_N}^{-1}((B_N)_{:,j}) = \sum_{\ell=1}^{\lceil \log_2 N\rceil} 2^{\ell-1}(B_N)_{\ell,j}. \tag{5.4}$$

Note that B_N is effectively the parity check matrix for a Hamming code with an additional row of 1's appended at the top, and so $f_{B_N}^{-1}$ effectively represents the efficient correction of a single error in the ECC context. In our setting this means that linear systems of the form $B_N\mathbf{x} = \mathbf{y}$ can be solved for all 1-sparse $\mathbf{x} = \gamma\mathbf{e}_j$ in $\mathcal{O}(\log N)$-time since for all such vectors we have

$$B_N\mathbf{x} = B_N(\gamma\mathbf{e}_j) = \gamma(B_N)_{:,j} = \mathbf{y}$$

so that

$$\mathbf{x} = y_0\mathbf{e}_{f_{B_N}^{-1}\left(\frac{1}{y_0}\mathbf{y}\right)}.$$

Of course, one is unlikely to encounter exactly 1-sparse vectors in many situations. Toward generalizing, suppose now instead that $\mathbf{x} = \gamma\mathbf{e}_j + \mathbf{n} \in \mathbb{C}^N$, and that we want to recover the correct $j \in [N]$ for $\mathbf{x}$ from noisy measurements of the form $B_N\mathbf{x}+\tilde{\mathbf{n}} \in \mathbb{C}^{1+\lceil \log_2 N\rceil}$. If $|\gamma| > \|\mathbf{n}\|_1 + 3\|\tilde{\mathbf{n}}\|_\infty$ it turns out that we can still recover j quickly from $\mathbf{y} = B_N\mathbf{x} + \tilde{\mathbf{n}}$ in this approximately 1-sparse setting using "bit testing". To see how, note that if the ℓ^{th}-bit of $j \in [N]$ in binary is 1 then

$$y_{\ell+1} = (B_N\mathbf{x}+\tilde{\mathbf{n}})_{\ell+1} = \langle (B_N\mathbf{x})_{\ell+1,:}, \mathbf{x}\rangle + \tilde{n}_{\ell+1} = \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} x_k + \tilde{n}_{\ell+1} = \gamma + \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} n_k + \tilde{n}_{\ell+1},$$

while

$$y_0 - y_{\ell+1} = \sum_{k\in[N]} x_k - \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} x_k + (\tilde{n}_0 - \tilde{n}_{\ell+1}) = \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =0}} x_k + (\tilde{n}_0 - \tilde{n}_{\ell+1}) = \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =0}} n_k + (\tilde{n}_0 - \tilde{n}_{\ell+1}).$$

Thus, $|y_{\ell+1}| > |y_0 - y_{\ell+1}|$ will hold if the ℓ^{th}-bit of of $j \in [N]$ is 1 since in that case

$$\begin{aligned}
|y_{\ell+1}| = \left| \gamma + \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} n_k + \tilde{n}_{\ell+1} \right| &\geq |\gamma| - \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} |n_k| - |\tilde{n}_{\ell+1}| \\
&> \|\mathbf{n}\|_1 + 3\|\tilde{\mathbf{n}}\|_\infty - \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =1}} |n_k| - |\tilde{n}_{\ell+1}| \\
&\geq \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =0}} |n_k| + 2\|\tilde{\mathbf{n}}\|_\infty \\
&\geq \left| \sum_{\substack{k\in[N] \text{ with} \\ \ell^{\text{th}} \text{ bit } =0}} n_k + (\tilde{n}_0 - \tilde{n}_{\ell+1}) \right| = |y_0 - y_{\ell+1}|.
\end{aligned}$$

Similarly, one can also see that $|y_0 - y_{\ell+1}| > |y_{\ell+1}|$ will hold if the ℓ^{th}-bit of j in binary is 0. Combining these observations we obtain the following lemma.

Lemma 5.1.29 (Bit Testing)**.** *Let $j \in [N]$, $\ell \in [\lceil \log_2 N \rceil]$, $\mathbf{n} \in \mathbb{C}^N$, $\tilde{\mathbf{n}} \in \mathbb{C}^{1+\lceil \log_2 N \rceil}$, and $\gamma \in \mathbb{C}$ satisfy $|\gamma| > 3(\|\mathbf{n}\|_1 + \|\tilde{\mathbf{n}}\|_\infty)$. Set $\mathbf{y} = B_N(\gamma \mathbf{e}_j + \mathbf{n}) + \tilde{\mathbf{n}}$. Then, the ℓ^{th}-bit of j when written in binary is 1 if and only if $|y_{\ell+1}| > |y_0 - y_{\ell+1}|$.*

We note here that the first known deterministic Sparse Fourier Transform (SFT) methods [29] utilized a different matrix than B_N with an alternate bit testing procedure that's instead based on the Chinese Remainder Theorem in order to prove an analogous result to Lemma 5.1.29. More generally, a useful variant of Lemma 5.1.29 can be proven for any matrix $A \in \{0,1\}^{m\times N}$ that has a rapidly computable inverse column map f_A^{-1}. For the sake of generality we next state a generalized bit testing result from which all such useful variants can be easily derived. It's proof is an straightforward generalization of the proof of Lemma 5.1.29.

Lemma 5.1.30 (Generalized Bit Testing)**.** *Let $A \in \{0,1\}^{m\times N}$ have an injective column map $f_A : [N] \to \{0,1\}^m$ defined by $f_A(j) = A_{:,j}$, and define $\tilde{A} \in \{0,1\}^{(m+1)\times N}$ to be A with an additional row of 1's appended at the top. Now let $j \in [N]$, $\ell \in [m]$, $\mathbf{n} \in \mathbb{C}^N$, $\tilde{\mathbf{n}} \in \mathbb{C}^{m+1}$, and $\gamma \in \mathbb{C}$ satisfy $|\gamma| > 3(\|\mathbf{n}\|_1 + \|\tilde{\mathbf{n}}\|_\infty)$. Set $\mathbf{y} = \tilde{A}(\gamma \mathbf{e}_j + \mathbf{n}) + \tilde{\mathbf{n}}$. Then, $(f_A(j))_\ell = A_{\ell,j} = 1$ if and only if $|y_{\ell+1}| > |y_0 - y_{\ell+1}|$.*

We are now prepared to present what will ultimately serve as our algorithm for rapidly finding small sets $S \supseteq \mathcal{C}_{s,\beta}(\mathbf{x}, \mathbf{n})$. When combined with Algorithm 23 it will provide us with a sublinear-time compressive sensing algorithm.

5.1.6 Support Recovery: Rapidly Finding a Good Set $S \subset [N]$

In this section $A \in \{0,1\}^{m_1 \times N}$ will always represent a matrix for which the function $f_A : [N] \to \{0,1\}^{m_1}$ defined by $f(j) = A_{:,j}$ is injective. As a consequence, we note that $f_A^{-1} : \{\text{columns of } A\} \subset \{0,1\}^{m_1} \to [N]$ will always exist. For any such matrix A, we will also let $\tilde{A} \in \{0,1\}^{(m_1+1)\times N}$ denote A with an additional first row of 1's appended to its top. Furthermore, in this section we will generally assume that $M \in \{0,1\}^{m_2 \times N}$ is (K,α)-coherent with $s \in [1, K/\alpha] \cap \mathbb{N}$ and $K > 4\alpha s$ so that M is also majority (s,K)-reconstructing for all $\mathbf{x} \in \mathbb{C}^N$ by Theorem 5.1.27. In that case we note that the cardinality of B_n in (5.1) will exceed $K/2$ for all $n \in [N]$ and $\mathbf{x} \in \mathbb{C}^N$.

Before we can present our support recovery algorithm we will also need some additional notation.

Definition 5.1.31 (Khatri–Rao Product)**.** *Let $B \in \mathbb{C}^{m\times N}$ and $C \in \mathbb{C}^{p\times N}$. Their Khatri–Rao product, $B \odot C \in \mathbb{C}^{mp\times N}$, is defined as*

$$B \odot C = \begin{pmatrix} B_{0,0}C_{:,0} & B_{0,1}C_{:,1} & \dots & B_{0,N-1}C_{:,N-1} \\ B_{1,0}C_{:,0} & B_{1,1}C_{:,1} & \dots & B_{1,N-1}C_{:,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m-1,0}C_{:,0} & B_{m-1,1}C_{:,1} & \dots & B_{N-1,N-1}C_{:,N-1} \end{pmatrix}.$$

More precisely, for any $j \in [mp]$ we note that we may uniquely write $j = qm + r$ for some $q \in [p]$ and $r \in [m]$. The $(j,k) \in [mp] \times [N]$ entry of $B \odot C$ is then $(B \odot C)_{j,k} = B_{r,k}C_{q,k}$.

Looking at Definition 5.1.31 we note that somewhat conveniently we also have

$$B \odot C = \begin{pmatrix} B_{0,:} \odot C \\ B_{1,:} \odot C \\ \vdots \\ B_{m-1,:} \odot C \end{pmatrix}. \tag{5.5}$$

In our support recovery algorithm below, Algorithm 24, we use measurements of the form $\mathbf{y} = \left(M \odot \tilde{A}\right)\mathbf{x}$. Choose any $r \in [m_2]$ you like. Due to (5.5) and the row of 1's always present in $\tilde{A}$, we note that any such $\mathbf{y}$ will always contain both $M\mathbf{x}$ and $\mathbf{y}_r := \left(M_{r,:} \odot \tilde{A}\right)\mathbf{x} = \tilde{A}(M_{r,:} \odot \mathbf{x})$ as subvectors. This fact is used heavily in Algorithm 24.

Algorithm 24 Support Recovery, SR : function pointer $\times\ \mathbb{C}^{(m_1+1)m_2} \times [m_2] \to \mathcal{P}([N])$.

INPUT: Pointer to $\tilde{f}_A^{-1}$ in line 10, $\mathbf{y} = \left(M \odot \tilde{A}\right)\mathbf{x} + \mathbf{n}$, K. Here $A \in \{0,1\}^{m_1 \times N}$ is as in the first paragraph of Section 5.1.6, and $M \in \{0,1\}^{m_2 \times N}$ is majority (s, K)-reconstructing (see Theorem 5.1.32).
OUTPUT: $S \subseteq [N]$

1: **for** $r \in [m_2]$ **do**
2: Locate $\mathbf{y}_r = \tilde{A}(M_{r,:} \odot \mathbf{x}) + \tilde{\mathbf{n}}_r \in \mathbb{C}^{m_1+1}$ as a subvector of $\mathbf{y} \in \mathbb{C}^{(m_1+1)m_2}$.
3: Initialize $\mathbf{z} \leftarrow \mathbf{0} \in \{0,1\}^{m_1}$.
4: **for** $\ell \in [m_1]$ **do**
5: **if** $\left|(\mathbf{y}_r)_{\ell+1}\right| > \left|(\mathbf{y}_r)_0 - (\mathbf{y}_r)_{\ell+1}\right|$ **then**
6: $z_\ell \leftarrow 1$.
7: **end if**
8: **end for**
9: *#In line 10,* $\tilde{f}_A^{-1} : \{0,1\}^{m_1} \to \mathbb{N}$ *can be any desired extension of* $f_A^{-1} :$ {columns of A} $\to [N]$ *#one likes to all of* $\{0,1\}^{m_1}$ *as long as it satisfies* $\tilde{f}_A^{-1}(A_{:,j}) = f_A^{-1}(A_{:,j}) = j\ \forall j \in [N]$.
10: $j_r \leftarrow \tilde{f}_A^{-1}(\mathbf{z})$.
11: **end for**
12: Initialize $S \leftarrow \emptyset$
13: **for** $k \in [m_2]$ **do**
14: **if** j_k appears in the sequence $\{j_r\}_{r\in[m_2]}$ created in line 10 more than $K/2$ times **then**
15: $S \leftarrow S \cup \{j_k\}$
16: **end if**
17: **end for**
18: Output $S \cap [N]$

Complexity Analysis

The runtime of Algorithm 24 can be accounted for as follows for $A \in \{0,1\}^{m_1 \times N}$ and $M \in \{0,1\}^{m_2 \times N}$:

- Note that $\mathbf{y}_r$ in line 2 can be located in $\mathcal{O}(m_1)$-time for each $r \in [m_2]$ using the structure of the Khatri–Rao product. Similarly, the inner "for loop" lines 4 – 8 together with its initialization step in line 3 can also be performed in $\mathcal{O}(m_1)$-time for each $r \in [m_2]$. Finally, assuming that $\tilde{f}_A^{-1} : \{0,1\}^{m_1} \to \mathbb{N}$ in line 10 can always be evaluated with at most T_A flops (i.e., in $\mathcal{O}(T_A)$-time), one can see that the entire outer "for loop" lines 1 – 11 can be performed in $\mathcal{O}(m_2 T_A + m_2 m_1)$-time.

- The final "for loop" together with its initialization step in lines 12 – 17 can be implemented by, e.g., (i) sorting the sequence $\{j_r\}_{r\in[m_2]}$, followed by (ii) reading

through the sorted sequence once while counting the lengths of strings of repeated elements (listing those repeated $> K/2$ times). Sorting the sequence $\{j_r\}_{r\in[m_2]}$ can be done in $\mathcal{O}(m_2 \log m_2)$-time using, e.g., merge sort [36]. Reading through the sorted sequence to find all elements repeated more than $K/2$ times can then be done in $\mathcal{O}(m_2)$-time. Thus, the total runtime of lines 12 – 17 is $\mathcal{O}(m_2 \log m_2)$.

- Line 18 can be implemented in $\mathcal{O}(m_2/K)$-time by simply reading through S without outputting elements not in $[N]$.

Therefore, the total runtime/flop count of Algorithm 24 is $\mathcal{O}\left(m_2(T_A + m_1 + \log m_2)\right)$. Furthermore, note that m_1 must be $\Omega(\log m_2)$ in order for f_A to be injective.[1] Thus, the algorithm's runtime is linear in the size of $\mathbf{y}$ if $\tilde{f}_A^{-1} : \{0,1\}^{m_1} \to \mathbb{N}$ in line 10 can be evaluated in $\mathcal{O}(m_1)$-time. Looking at (5.4) one can see that this will indeed be the case when, e.g., $\tilde{A}$ is chosen to be the N^{th} bit testing matrix. In particular, if $\tilde{A} = B_N$ then Algorithm 24 will be $\mathcal{O}(m_2 \log N)$-time. Finally, it's also worth mentioning that Algorithm 24 is trivially parallelizable since, e.g., lines 2 – 10 can all be run in parallel for different $r \in [m_2]$. Furthermore, lines 12 – 18 can also be parallelized by, e.g., (i) using efficient parallel sorting methods to sort $\{j_r\}_{r\in[m_2]}$ after it's formed (see, e.g., [43]), and then by (ii) counting the number of times each element is repeated in the sorted sequence (and then outputting as appropriate) in parallel for each $k \in [m_2]$.

Theoretical Guarantees

The following Theorem guarantees that Algorithm 24 will recover a relatively small set S that contains all of $\mathcal{C}_{s,3}(\mathbf{x}, \mathbf{n})$ as desired.

Theorem 5.1.32. *Let $\mathbf{x} \in \mathbb{C}^N$ and define $\tilde{\mathbf{x}} \in [0,\infty)^N$ to be such that $\tilde{x}_n = |x_n|$ for all $n \in [N]$. Suppose further that $M \in \{0,1\}^{m_2 \times N}$ is majority (s,K)-reconstructing for $\tilde{\mathbf{x}}$, $A \in \{0,1\}^{m_1 \times N}$ is as in the first paragraph of Section 5.1.6, and $\mathbf{n} \in \mathbb{C}^{(m_1+1)m_2}$. Set $\mathbf{y} = \left(M \odot \tilde{A}\right)\mathbf{x} + \mathbf{n}$. Then, $\mathrm{SR}\left(\tilde{f}_A^{-1}, \mathbf{y}, K\right)$ will output a set $S \subseteq [N]$ of cardinality $|S| < \frac{2m_2}{K}$ satisfying*

$$\mathcal{C}_{s,3}(\mathbf{x},\mathbf{n}) = \left\{ n \in [N] : |x_n| > 3\left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right) \right\} \subseteq S.$$

Proof. The cardinality bound $|S| < \frac{2m_2}{K}$ is ensured by lines 12 – 17 of Algorithm 24, and $S \subseteq [N]$ is ensured by line 18. Thus, the remainder of the proof will be dedicated to proving that $\mathcal{C}_{s,3}(\mathbf{x},\mathbf{n}) \subseteq S$. Toward that end, let $k \in \mathcal{C}_{s,3} \subseteq [N]$, and suppose that $j \in B_k$ for $\tilde{\mathbf{x}}$ in (5.1) so that

$$\left| \left(M(k)\tilde{\mathbf{x}}\right)_j - \tilde{x}_k \right| \leq \frac{\|\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_s\|_1}{s} = \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s}. \tag{5.6}$$

[1] Here we assume $m_2 \leq N$ since, if not, none of the techniques discussed herein should be used anyway.

Note that every $j \in [K]$ satisfying (5.6) maps injectively to a unique $j' \in [m_2]$ satisfying both $M_{j',k} = 1$ and $\left\| \sum_{\ell \neq k} \left(M_{j',:} \odot \mathbf{x} \right)_\ell \mathbf{e}_\ell \right\|_1 = \left| \sum_{\ell \neq k} \left(M_{j',:} \odot \tilde{\mathbf{x}} \right)_\ell \right| \leq \frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s}$. Let $\mathbf{y}_{j'} = \tilde{A}(M_{j',:} \odot \mathbf{x}) + \tilde{\mathbf{n}}_{j'}$ be the subvector of $\mathbf{y}$ found for $r = j'$ in line 2 of Algorithm 24. We can see that $M_{j',:} \odot \mathbf{x} = \gamma \mathbf{e}_k + \mathbf{n}'$ where $\mathbf{n}' = \sum_{\ell \neq k} \left(M_{j',:} \odot \mathbf{x} \right)_\ell \mathbf{e}_\ell$ and $|\gamma| = |x_k| > 3 \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right) \geq 3 \left(\|\mathbf{n}'\|_1 + \left\|\tilde{\mathbf{n}}_{j'}\right\|_\infty \right)$. As a consequence, when $r = j'$ Lemma 5.1.30 guarantees that lines 4 – 8 of Algorithm 24 will recover $\mathbf{z} = A_{:,k}$ so that k is added to the sequence created by line 10.

Moreover, the fact that M is majority (s, K)-reconstructing for $\tilde{\mathbf{x}}$ guarantees that k must appear in the sequence created by line 10 more than $K/2$ times since more than $K/2$ different $j \in [K]$ must satisfy (5.6). Hence, k will be added to S in line 15 of Algorithm 24. Finally, the fact that $k \in [N]$ guarantees that it won't then be removed from S later in line 18. □

In order to obtain the main result of this paper it will be convenient to apply Theorem 5.1.32 with the matrix A chosen as below.

Definition 5.1.33 (β-Augmented Bit Testing Matrices)**.** *Let $\beta \in \mathbb{N}$. The β-Augmented Bit Testing Matrix $B_N^\beta \in \{0,1\}^{(\beta + \lceil \log_2 N \rceil) \times N}$, is the matrix whose j^{th}-column $\forall j \in [N]$ consists of β 1's followed by j written in binary.*

The following result is a simple corollary of Theorem 5.1.32.

Corollary 5.1.34. *Choose an integer $\beta > 0$. Let $\mathbf{x} \in \mathbb{C}^N$ and define $\tilde{\mathbf{x}} \in [0, \infty)^N$ to be such that $\tilde{x}_n = |x_n|$ for all $n \in [N]$. Suppose further that $M \in \{0,1\}^{m_2 \times N}$ is majority (s, K)-reconstructing for $\tilde{\mathbf{x}}$, let $B_N^\beta \in \{0,1\}^{(\beta + \lceil \log_2 N \rceil) \times N}$ be as in Definition 5.1.33, and $\mathbf{n} \in \mathbb{C}^{(\beta + \lceil \log_2 N \rceil) m_2}$. Set $\mathbf{y} = \left(M \odot B_N^\beta \right) \mathbf{x} + \mathbf{n}$. Then,* $\text{SR}\left(\tilde{f}^{-1}_{B_N^{\beta-1}}, \mathbf{y}, K \right)$ *will output a set $S \subseteq [N]$ of cardinality $|S| < \frac{2m_2}{K}$ satisfying*

$$\mathcal{C}_{s,3}(\mathbf{x}, \mathbf{n}) = \left\{ n \in [N] : |x_n| > 3 \left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{s} + \|\mathbf{n}\|_\infty \right) \right\} \subseteq S.$$

Furthermore, $\text{SR}\left(\tilde{f}^{-1}_{B_N^{\beta-1}}, \mathbf{y}, K \right)$ *can be executed in $\mathcal{O}\left(m_2(\beta + \log N)\right)$-time.*

Proof. Apply Theorem 5.1.32 with $A = B_N^{\beta-1}$. Noting that $\tilde{A} = B_N^\beta$ is a trivial extension of B_N in this case, one can see that line 10 of Algorithm 24 can be implemented to run in $\mathcal{O}(\beta + \log N)$-time. The stated runtime complexity now follows from Section 5.1.6. □

We are now prepared to provide some applicable sublinear-time compressive sensing results which will later be applied to approximate compressible eigenvectors.

5.1.7 Our Final Sublinear-Time Recovery Result

The following recovery result will utilize $A = B_N^{\beta-1}$ for some integer $\beta > 1$ in Algorithm 24 above. We hasten to add here, however, that those wishing to, e.g., develop sparse Fourier transform variants will need to choose different matrices A. The recovery procedure we will analyze follows.

Algorithm 25 Fast Compressive Sensing, FastCS : $\mathbb{N} \times [N] \times [m'] \times \mathbb{C}^{(\beta+\lceil \log_2 N \rceil)m'} \to \mathbb{C}^N$.

INPUT: $\beta \in \mathbb{N}$, $s \in [N]$, $K \in [m']$, $\mathbf{y} = \left(M \odot B_N^{\beta}\right)\mathbf{x} + \mathbf{n} \in \mathbb{C}^{(\beta+\lceil \log_2 N \rceil)m'}$. Here $M \in \{0,1\}^{m' \times N}$ is majority (s,K)-reconstructing (see Theorem 5.1.35).
OUTPUT: $\mathbf{z}|_{\tilde{S}} \in (\mathbb{C} \times [N])^{\leq 2s}$, encoding a sparse vector $\mathbf{z} \in \mathbb{C}^N$

1: Find support set $S' \leftarrow \mathrm{SR}\left(\tilde{f}^{-1}_{B_N^{\beta-1}}, \mathbf{y}, K\right) \subseteq [N]$.
2: Locate $\mathbf{y}' = M\mathbf{x} + \mathbf{n}' \in \mathbb{C}^{m'}$ as a subvector of $\mathbf{y} \in \mathbb{C}^{(\beta+\lceil \log_2 N \rceil)m'}$.
3: Output $\mathrm{MR}(\mathbf{y}', S', s)$ as a sparse approximation to $\mathbf{x}$.

The following theorem provides a best s-term approximation guarantee [13] for Algorithm 25.

Theorem 5.1.35. *Let $\beta \in [1,\infty) \cap \mathbb{Z}$, $s \in [N]$, $\mathbf{x} \in \mathbb{C}^N$, and define $\tilde{\mathbf{x}} \in [0,\infty)^N$ to be such that $\tilde{x}_n = |x_n|$ for all $n \in [N]$. Suppose that $M \in \{0,1\}^{m' \times N}$ is majority (s,K)-reconstructing for $\tilde{\mathbf{x}}$, let $B_N^{\beta} \in \{0,1\}^{(\beta+\lceil \log_2 N \rceil) \times N}$ be as in Definition 5.1.33, and $\mathbf{n} \in \mathbb{C}^{(\beta+\lceil \log_2 N \rceil)m'}$. Set $\mathbf{y} = \left(M \odot B_N^{\beta}\right)\mathbf{x} + \mathbf{n}$. Then,*

$$\|\mathbf{x} - \mathrm{FastCS}(\beta, s, K, \mathbf{y})\|_2 \leq \|\mathbf{x} - \mathbf{x}_{2s}\|_2 + 6(1+\sqrt{2})\left(\frac{\|\mathbf{x} - \mathbf{x}_s\|_1}{\sqrt{s}} + \sqrt{s}\|\mathbf{n}\|_\infty\right). \tag{5.7}$$

Furthermore, $\mathrm{FastCS}(\beta, s, K, \mathbf{y})$ *can be executed in $\mathcal{O}\left(m'(\beta + \log N)\right)$-time whenever the K rows of $M(n)$ can be identified in $\mathcal{O}(K)$-time for any given $n \in [N]$.*

Proof. We know that $S' \subseteq [N]$ contains $\mathcal{C}_{s,3}(\mathbf{x}, \mathbf{n})$ by Corollary 5.1.34. Furthermore, we know that M is also majority (s,K)-reconstructing for $\mathbf{x}$ by Lemma 5.1.19. Hence, (5.7) will hold by Theorem 5.1.20.

Considering the quoted runtime complexity for Algorithm 25, we note that Line 1 will run in $\mathcal{O}\left(m'(\beta + \log N)\right)$-time and produce a set S' satisfying $|S'| < \frac{2m'}{K}$ by Corollary 5.1.34. Similarly, the subvector $\mathbf{y}'$ in Line 2 can always be located in $\mathcal{O}\left(m'(\beta + \log N)\right)$-time. Finally, recalling the discussion in Section 5.1.3, Line 3 will run in

$$\mathcal{O}\left(m' + |S'| \cdot \max\left(K, \log|S'|\right)\right) = \mathcal{O}\left(m' \cdot \max\left(1, \frac{\log(m'/K)}{K}\right)\right)$$

time whenever the K rows of $M(n)$ can be identified in $\mathcal{O}(K)$-time for any given $n \in [N]$. The result now follows after noting that m' will be $\mathcal{O}(N)$ in all non-trivial applications of these results. $\square$

Bibliography

[1]

[2]

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468, 2006.

[4] J. Bailey, M. A. Iwen, and C. V. Spencer. On the design of deterministic matrices for fast recovery of fourier compressible functions. *SIAM Journal on Matrix Analysis and Applications*, 33(1):263–289, 2012.

[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] G. E. Box and M. E. Muller. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 29(2):610–611, 1958.

[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] 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.

[12] B. A. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2, 2000.

[13] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *Journal of the American mathematical society*, 22(1):211–231, 2009.

[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] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery.

[17] J. W. Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[18] R. A. DeVore. Deterministic constructions of compressed sensing matrices. *Journal of Complexity*, 23(4):918 – 925, 2007.

[19] G. B. Folland. *Fourier analysis and its applications*. American Mathematical Soc., 2009.

[20] S. Foucart. *Mathematical pictures at a data science exhibition*. Cambridge University Press, 2022.

[21] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser/Springer, New York, 2013.

[22] M. Frigo and S. Johnson. The design and implementation of fftw3. *Proceedings of IEEE 93 (2)*, pages 216–231, 2005.

[23] S. R. Garcia and R. A. Horn. *Matrix Mathematics: A Second Course in Linear Algebra*. Cambridge University Press, 2023.

[24] V. Guillemin and A. Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.

[25] M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the fast fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, 1984.

[26] R. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[27] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2013.

[28] M. Iwen. Compressed sensing with sparse binary matrices: Instance optimal error guarantees in near-optimal time. *Journal of Complexity*, 30(1):1 – 15, 2014.

[29] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 08, pages 20–29, USA, 2008. Society for Industrial and Applied Mathematics.

[30] M. A. Iwen. Simple deterministically constructible rip matrices with sublinear fourier sampling requirements. In *2009 43rd Annual Conference on Information Sciences and Systems*, pages 870–875, March 2009.

[31] 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.

[32] 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.

[33] B. S. Kashin. The diameters of octahedra. *Uspehi Mat. Nauk*, 30(4(184)):251–252, 1975.

[34] D. R. Kincaid and E. W. Cheney. *Numerical analysis: mathematics of scientific computing*, volume 2. Brooks/Cole Pacific Grove, CA, 2002.

[35] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[36] M. A. Kronrod. An optimal ordering algorithm without a field of operation. *Dokl. Akad. Nauk SSSR*, 186:1256–1258, 1969.

[37] I. Lau and J. Jedwab. Construction of binary matrices for near-optimal compressed sensing. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1612–1617. IEEE, 2021.

[38] J. E. Marsden and A. Tromba. *Vector Calculus, Sixth Edition*. W.H. Freeman and Company, 2012.

[39] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 154–157, New York, NY, USA, 2006. Association for Computing Machinery.

[40] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[41] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An introduction to the theory of numbers*. John Wiley & Sons, 1991.

[42] C. H. Papadimitriou. *Computational Complexity*. Addiaon-Wesley Publishing Company, Inc., 1994.

[43] D. Pasetto and A. Akhriev. A comparative study of parallel sort algorithms. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, OOPSLA '11, page 203–204, New York, NY, USA, 2011. Association for Computing Machinery.

[44] G. Plonka, D. Potts, G. Steidl, and M. Tasche. *Numerical Fourier Analysis*. Springer, 2018.

[45] 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.

[46] H. L. Royden and P. Fitzpatrick. *Real Analysis, Fourth Edition*. Pearson Education, Inc., 2010.

[47] 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.

[48] G. Stewart and J.-g. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing/Academic Press, Inc, 1990.

[49] G. W. Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[50] V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[51] L. N. Trefethen and D. Bau. *Numerical linear algebra*. SIAM, 2022.

[52] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018.

[53] 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.