# Detection, Estimation, and Learning

Prof. Matthew Nokleby
Wayne State University

# Preface

This monograph began as a set of lecture notes to accompany the main textbook for my Detection and Estimation course, first taught in the Electrical and Computer Engineering Department at Wayne State University in Fall 2016. That semester, the notes took on a life of their own: students found the notes rather accessible relative to the assigned text, and before long the notes became the students' primary reference. Writing in a hurry over nights and weekends, by the end of the semester I had produced the first version of these notes, which adequately presented the material in a standard, one-semester instantiation of detection and estimation: hypothesis testing, ROC curves, maximum-likelihood and Bayes estimation, and the Kalman filter.

By the time I taught the course a second time, I had identified a need for ECE grad students to be exposed to the basics of machine learning, which, after all, are intimately related to the statistical signal processing techniques at the heart of detection and estimation. This time, I incorporated ML approaches that one can derive "from scratch" using principles from detection and estimation theory: linear regression and classifiers, feature extraction via principal components and $k$-means clustering, and sparse coding and regression. The result is a unified approach that puts ideas often studied disparately—signal detection and filtering (electrical engineering), statistical machine learning (statistics and computer science), and image classification processing (computer science and electrical engineering)—under a single umbrella.

I also found that students benefit from "hands-on" interaction with the course material. It's one thing to prove the asymptotic efficiency of the maximum-likelihood estimator, and it's quite another to see estimator performance improve on real data. To meet this need, I developed a handful of hands-on examples to accompany the notes. These are written in Python, presented as Jupyter Notebooks, and can be found at my GitHub page (http://github.com/docnok/detection-estimation-learning). These examples make liberal use of public datasets that can be accessed through the `scikit-learn` library of machine learning tools for Python. These example are intended for in-class demonstration: I have found it useful to step through these examples in class after working through concepts in theory.

Owing to a rushed writing schedule—ensured by the demands on the time of tenure-track faculty—there surely remain numerous errors in the text. I welcome corrections. Another major shortfall is the lack of proofs for many of the results. This omission is primarily due to the fact that this monograph began life as an accompaniment to a text, where students could easily find formal proofs. I am a big believer in the pedagogical value of proofs, and they will be included in future editions. It's also worth noting that I have taken a somewhat idiosyncratic approach to the content. For example, I view sparse learning and regularized linear regression through a Bayesian lens, where regularizers emerge directly from the imposition of a prior. This approach is, in my experience, not the dominant one, but I found it to give the cleanest unification between machine learning ideas and estimation theory.

Finally, I have drawn on a multitude of sources in preparing this document. The overall structure follows *An Introduction to Signal Detection and Estimation* by Poor. Many of the discussions and intutions are adapted from *Mathematical Methods and Algorithms for Signal Processing* by Moon and Stirling. Many of the signal processing examples are adapted from *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory* by Kay. The discussion for the machine learning section, especially the linear regression content, is inspired by *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman. The introductory material on probability theory is inspired by *Information Theory, Inference, and Learning Algorithms* by MacKay, and the introductory material on optimization theory is inspired by *Deep Learning* by Goodfellow, Bengio, and Courville. All errors remain my own.

# Chapter 1

# Introduction

The goal of this course is to introduce you to two related topics in modern signal processing and machine learning: **detection** and **estimation**. These are "old" terms, left over from (say) radar engineering, but they are fundamental to modern data science. Anytime you are given a pile of data and asked to learn something from it, you will need the ideas and techniques we'll cover in this class—whether you know it or not!

In a formal sense, the main distinction between detection and estimation is the size of the hypothesis space. In detection, the hypothesis space is discrete, or even finite. We ask questions like *is an object present?* or *which codeword was sent over the channel?*. On the other hand, in estimation the hypothesis space is continuous. We ask questions like *what are the pixel values of a corrupted image?* or *where in three-dimensional space is an object located?* This distinction will occasionally seem arbitrary, and there are real-world cases that blur the distinction between detection and estimation. But this terminology is standard in the literature and among practitioners, and we'll stick with it.

To get a better sense of the subject matter, here are a few concrete problems that we'll look at. First, a few detection problems:

- **Object detection and recognition:** You shine a radar out over the ocean; is there an aircraft out there? Which kind of aircraft is it? Or, for more locally relevant: You shine a LIDAR into the street; is there an obstacle? Is it a pedestrian, a cyclist, or an automobile? Deciding whether or not an object is present is called **object detection**. Figuring out which *kind* of object is present is called **object recognition**. In the first case, the hypothesis space is binary: either there is a target or there isn't, and we need to choose between the two options. In the second case, the hypothesis space is larger: we need to choose among all possible types of objects, which is a trickier task.
- **Decoding in digital communications:** Your mobile phone receives a packet from an access point; how does it know which bits were sent? This is the **decoding** problem. The hypothesis space here can be rather large. If a packet contains $k$ bits, there are $2^k$ total packets possible. A challenge solved by the phones in all our pockets is to develop a decoder that can quickly decode a packet from a noisy transmission.

Next, a few estimation problems.

- **Image denoising:** If you look at an image that's corrupted by noise or physical damage, you may have an intuitive sense of how to correct it. You might follow contours in the clean part of the image, or mimic colors and patterns in nearby pixels. Doing so in an automated fashion is called *denoising*, and it has a long history in the signal processing literature. We will look at simple approaches later in the semester.

- **Recommender systems:** Suppose you have rated a set of movies and TV shows on a streaming video service. Can the service predict how much you will like other items in its library? Finding new items for you to consume, and predicting how much you will like them, is at the heart of **recommender systems**. A wide variety of techniques have been proposed and put into practice; making them work better is an active area of research.

There is a third area, which technically falls under estimation, but which is sufficiently distinct that we will give it its own name: **tracking**. When NASA tracks objects in space, it has two sources of information. First, it has its (radar, etc.) sensor data, which gives direct information about the position and velocity of the object. Second, it knows that the object is subject to the laws of physics. If we know the object's location and velocity right now, we can follow basic dynamics and predict the object's state in a coming moment. How do we combine that information? In the 1960s, researchers developed what's now called the *Kalman filter*, and which has become the standard approach to object tracking. In addition to widespread use today, the Kalman filter's claim to fame is its use on the lunar module of the Apollo program. We'll look at the Kalman filter in the last unit of this course.

These are the problems; how will we solve them? Modern machine learning boils down to three main tools, which we'll use a lot: (1) **probability theory**, (2) **linear algebra**, and (3) **optimization theory**. Using probability theory, we impose a **statistical model** on whatever we're trying to detect, estimate, or track. Our data are usually high-dimensional, so we represent them using **vectors and matrices**. Finally, we use optimization theory to **find the best solution** to our problem according to the statistical model we've imposed. These three tools can seem abstract at first, but there's no getting around learning them if you want to do data science in the 21st century.

This approach leads to the algorithms that are mainstays of signal processing and machine learning, and which you may have heard of in your research: maximum-likelihood estimation, linear/logistic regression, principal components analysis, and more. As we cover the underlying principles of detection and estimation, we'll take detours in which we examine these techniques, deriving them from first principles and trying them out on real-world data. At the course website, you can find Python scripts that illustrate these concepts and approaches.

Most of what you'll need in this course you've seen before, but since the three tools mentioned above are so crucial, we'll spend a chapter reviewing the concepts in detail.

# Chapter 2

# Review of Probability, Linear Algebra, and Optimization

## 2.1 Review of Probability Theory

The raw material we work with in data science—that is, the data we observe and the phenomena we are trying to model—are unavoidably uncertain. This uncertainty is due to many sources: noise on our sensors, incomplete observation, and also inherent variability of the system we are trying to model. To handle this uncertainty in a principled way, we'll model the data and phenomena using probability theory.

In this course, we'll take the **random variable**, denoted in capital letters ($X$,$Y$, etc.) as the main object of study. These random variables will represent the data we observe and the information we are trying to extract from that data. Formally, a random variable $X$ takes values in some set $\Gamma$. The set $\Gamma$ is the set of values that $X$ can take. For example, if $X$ is the result of a coin flip, we have $\Gamma = \{h, t\}$ for "heads" and "tails." Or, if $X$ is a letter in a text string, we have $\Gamma = \{a, b, \ldots, z\}$, ignoring cases and special characters.

We'll make the distinction between discrete random variables—for which $\Gamma$ is a finite or at least countable set—and continuous random variables, for which $\Gamma$ is a region of Euclidean space. In the coin flip and letter examples in the previous paragraph, $X$ is discrete. But many phenomena are continuous-valued. If $X$ is the height of an individual (measured in meters), $\Gamma = (0, \infty)$. Or if $X$ is the error on a sensor that measures the distance to an object, $\Gamma = (\infty, \infty)$, since the measurement error can be negative and (in principle!) be any real number.

To fully describe a random variable, we need to define its **probabilities**: what is the probability that $X$ takes on a particular value in $\Gamma$? To do so, we'll define the **distribution** of $X$. The distribution of a discrete random variable is a bit simpler to work with, but we'll more often work with continuous random variables. We'll build up our probabilistic intuition with respect to discrete random variables first, and then we'll move on to continuous random variables.

### 2.1.1 Discrete Random Variables

The distribution of a discrete random variable $X$ is described by its **probability mass function** (PMF), denoted $p_X(x)$:

$$p_X(x) = \Pr(X = x). \tag{2.1}$$

The expression $\Pr(X = x)$ represents the probability that the random variable $X$ takes on the hypothetical value $x \in \Gamma$. For example, a fair coin has a uniform distribution over $\Gamma = \{h, t\}$, meaning that

$$p_X(x) = \begin{cases} 1/2, & x = h \\ 1/2, & x = t \end{cases}. \tag{2.2}$$

A biased (or "unfair") coin will have different values for $p_X(0)$ and $p_X(1)$. When it is clear from context, we will drop the subscript and write $p(x)$. If $X$ is the letter in a string of English text, we can model the distribution of $X$ by using the relative frequency of each letter. I've taken letter frequencies from Wikipedia [1], giving the following PMF:

$$p(x) = \begin{cases} 0.08167, & x = a \\ 0.01492, & x = b \\ 0.02782, & x = c \\ \vdots \\ 0.00074, & x = z \end{cases}. \tag{2.3}$$

The probability mass function satisfies two crucial properties. First, it is **nonnegative**:

$$p_X(x) \geq 0, \tag{2.4}$$

because a negative probability is meaningless. Second, it is normalized:

$$\sum_{x \in \Gamma} p_X(x) = 1, \tag{2.5}$$

because the probability that *something* happens is one. These properties are useful sanity checks: if after computing a probability you have a negative number, or something that sums up to more than one, you can be sure that there is an error in your calculation (or in your code)!

#### 2.1.1.1 Probability of Events

An important notion is the probability of an **event**. Informally, an event is just something that might happen that we care about. If $X$ is a letter, we may not care about its exact value, but whether or not its a vowel or a consonant. Formally, we define an event as a **subset** $E \subset \Gamma$. The probability of the event $E$ is the probability that $X$ takes a value in that set $E$:

$$\Pr(E) = \Pr(X \in E). \tag{2.6}$$

For example, if $E = \{a, e, i, o, u\}$, then $E$ is the event that $X$ is a vowel. We can calculate the probability of an event using the PMF. Specifically, the probability of the event $E$, or the probability that $X \in E$, is the sum of the PMF over $E$:

$$\Pr(X \in E) = \sum_{x \in E} p_X(x). \tag{2.7}$$

[1] https://en.wikipedia.org/wiki/Letter_frequency

This is true because of one of the basic rules of probability: **the probability of disjoint events add up**. Adding up the probability of each of the "singleton" events in $E$, get the probability that $X \in E$. Two important results come from this fact. First,

$$\Pr(X \in \Gamma) = 1, \Pr(X \in \emptyset) = 0, \tag{2.8}$$

where $\emptyset$ is the empty set. The event $E = \Gamma$ is simply the event in which *something* happens, and it has probability one. The event $E = \emptyset$ is the event in which *nothing* happens, and it has probability zero! Second,

$$\Pr(E^c) = 1 - \Pr(E), \tag{2.9}$$

where $E^c$ is the **complement** of $E$. If the probability that $E$ happens is $\Pr(E)$, the probability that $E$ *doesn't* happen is $1 - \Pr(E)$.

#### 2.1.1.2 Examples of Discrete Distributions

Let's consider two important discrete distributions. The first is the **Bernoulli distribution**, for which $\Gamma = \{0, 1\}$ and the probabilities correspond to a weighted coin. We write $X \sim \mathcal{B}(p)$ if its probability mass function is

$$p(x) = \begin{cases} 1 - p, & x = 0 \\ p, & x = 1 \end{cases}. \tag{2.10}$$

A generalization of the Bernoulii distribution to random variables taking on more than two values is called the **categorical distribution**[2] Given a $k$-length vector $\mathbf{p}$ such that $\sum_{i=1}^{k} p_i = 1$, the categorical distribution has $\Gamma = \{1, \ldots, k\}$ and the pmf

$$p(x) = \begin{cases} p_1, & x = 1 \\ p_2, & x = 2 \\ \vdots & \\ p_3, & x = k \end{cases}. \tag{2.11}$$

The categorical distribution is just a way to refer in general to a discrete distribution over $k$ unique states.

The final example is the **uniform distribution**, for which $\Gamma = \{a, \ldots, b\}$ for integers such that $b > a$. We say that $X \sim \mathcal{U}(a, b)$ if the probability mass is evenly distributed over the interval:

$$p(x) = \frac{1}{b - a},\ a \leq x \leq b. \tag{2.12}$$

The uniform distribution is a special case of the categorical distribution.

### 2.1.2 Continuous Random Variables

The situation is more complicated for continuous random variables, where $\Gamma = \mathbb{R}$. In this case, it usually doesn't make sense to talk about the probability that $X$ takes on any one value. Instead, the probability of any event $E = \{x\}$ is (usually) zero! Consider the output of a random number

[2] In machine learning circles, some authors like to use the neologism "**multinoulii**" for the categorical distribution.

generator (like `rand()` in MATLAB or `np.random.uniform()` in NumPy) over the interval $[0, 1)$. The probability that $X$ takes on any specific value is zero, so we cannot write a probability mass function as we could in the discrete case.

Instead, we describe the distribution of $X$ in terms of the probability *density* around any point. We denote the **probability density function** (PDF) of $X$ as $p_X(x)$. Like the PMF, the probability density function is nonnegative. It also is normalized with respect to integration

$$\int_{-\infty}^{\infty} p_X(x)dx = 1. \tag{2.13}$$

However, it *is* possible that $p_X(x) > 1$.

The PDF is the analogue of the PMF for continuous random variables. But what does it mean? Instead of giving the probability of a specific value of $x$, the PDF allows us to compute the probability that $X$ lives inside a *region* of $\Gamma$. Specifically, we probability of any interval event $E = [a, b]$ is computed by *integrating* the PDF:

$$\Pr(E) = \int_a^b p_X(x)dx. \tag{2.14}$$

The PDF gives the *relative* probability of a slice of a region in $\Gamma$. Furthermore, the probability of any event $E$ is computed by integrating the density function over the set:

$$\Pr(\mathrm{E}) = \int_E p_X(x)dx. \tag{2.15}$$

The simplest example is the **uniform distribution** over the interval $\Gamma = [0, 1)$. Then, the PDF is

$$p_X(x) = \begin{cases} 0, & x < 0 \\ 1, & 0 \le x \le 1 \\ 0, x > 1 \end{cases}. \tag{2.16}$$

The PDF is constant, meaning that the *relative* probability of any slice along the interval $[0, 1)$ is the same. Finding the probability of an interval is easy in this case. For $E = [a, b]$ and $0 \le a < b < 1$, we have

$$\Pr(E) = \int_a^b dx = b - a. \tag{2.17}$$

Furthermore, the probability of any more complicated $E \subset [0, 1)$ is just the total width of the region.

A more principled way of defining the distribution of a continuous random variable is via the **cumulative distribution function** (CDF). The CDF of $X$, specifies the probability of intervals of the form $E = (-\infty, x]$:

$$F_X(x) = \Pr(X \le x). \tag{2.18}$$

For the uniform distribution, the CDF is

$$F_X(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}. \tag{2.19}$$

Like the PMF, the CDF is nonnegative. It also is an increasing function in $x$: as $x$ increases, the probability that $X$ lies in the interval $[-\infty, x]$ also increases. The CDF has the following limits:

$$\lim_{x \to -\infty} F_X(x) = 0, \qquad \lim_{x \to \infty} F_X(x) = 1. \tag{2.20}$$

Via the fundamental theorem of calculus, it's easy to see that the PDF is the derivative of the CDF:

$$p_X(x) = \frac{d}{dx} F_X(x). \tag{2.21}$$

As with the PMF, we will express the PDF as $p(x)$ when context makes it clear which density is meant.

#### 2.1.2.1 Examples of Continuous Distributions

The continuous **uniform distribution** is the continuous counterpart to the uniform distribution above. For a continuous random variable $X$, we write $X \sim \mathcal{U}(a, b)$ if the probability density function is

$$p(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & x < a \text{ or } x > b. \end{cases} \tag{2.22}$$

Perhaps the most important continuous distribution is the **normal** or **Gaussian distribution**. We say that $X \sim \mathcal{N}(\mu, \sigma^2)$ if its PDF is

$$\mathcal{N}(\mu, \sigma^2) = p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.23}$$

for mean value $\mu$ and variance $\sigma^2$. If $X$ is normally distributed, then it is centered around $\mu$ and the variance $\sigma^2$ describes how tightly $X$ clusters around the mean. When modeling noise or variability, the normal distribution is often the "default" choice. A signal processing interpretation of the Gaussian distribution is that the "true" signal is the mean $\mu$, and we observe a noisy version of the signal with average noise power $\sigma^2$.

The CDF of the normal distribution is sufficiently important that we will give it its own notation. Let $\Phi(x)$ denote the CDF of a *standard* normal distribution (i.e. with $\mu = 0$ and $\sigma^2 = 1$):

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right). \tag{2.24}$$

There is no closed-form expression for $\Phi(x)$. Other sources (including MATLAB and NumPy) use similar expressions derived from the CDF, including the Q-function and `erfc`.

Two more continuous distributions will be important. First is the **exponential distribution**, which has the PDF

$$p(x) = \lambda \exp(-\lambda x), \ x \geq 0, \tag{2.25}$$

for the *rate parameter* $\lambda > 0$. We say that $X \sim \text{Exp}(\lambda)$ if it has PDF $p(x)$. The exponential distribution is used to model the time between events, such as photons striking a photographic surface or arrivals of packets in a network. If such an event happens at a rate of $\lambda$ events per second, then $X \sim \text{Exp}(\lambda)$ is a good model for how many seconds pass between events.

A related distribution is the **Laplace distribution**, a two-sided version of the exponential distribution. We say that $X \sim \text{Laplace}(\mu, \gamma)$ if

$$p(x) = \frac{1}{2\gamma} \exp(-\frac{|x-\mu|}{\gamma}). \tag{2.26}$$

The Laplace distribution is used to model phenomena that are more concentrated around the mean than the Gaussian distribution.

### 2.1.3 Expectation

The **expected value** of the random variable $X$ is defined as

$$E[X] = \sum_{x \in \Gamma} x p_X(x) \tag{2.27}$$

if $X$ is a discrete random variable, and

$$E[X] = \int_{-\infty}^{\infty} x p_X(x) dx \tag{2.28}$$

if $X$ is continuous. Formally, the expected value $E[X]$ is the *centroid* of the density or mass function, and intuitively it represents the mean or average value of $X$. In the Gaussian case $X \sim \mathcal{N}(\mu, \sigma^2)$, $E[X] = \mu$, and we will use $\mu$ to denote the expectation of random variables regardless of their distribution.

A few examples. The expected value of a Bernoulli distribution is

$$E[X] = 0(1-p) + 1(p) = p, \tag{2.29}$$

and the expected value of the uniform distribution is

$$E[X] = \frac{1}{b-a}(a + \cdots + b) = \frac{a+b}{2}, \tag{2.30}$$

which is just the arithmetic mean of $a$ and $b$. By definition, the mean of a Gaussian distribution is $\mu$.

The expectation has a few important properties. First, expectation is *linear*. For a constant $a$ and for random variables $X$ and $Y$,

$$E[aX] = aE[X] \tag{2.31}$$

$$E[X+Y] = E[X] + E[Y]. \tag{2.32}$$

A crucial result is the **fundamental theorem of expectation**, which allows us to compute the expected value of any *function* of a random variable. For a function $f(x)$, the expectation $E[f(X)]$ is

$$E[f(X)] = \int_{-\infty}^{\infty} f(x) p_X(x) dx, \tag{2.33}$$

if $X$ is continuous. The expectation is the analogous sum if $X$ is discrete. In other words, we can just integrate $f(x)$ over the PDF in order to find its expected value.

An important special case is the **variance** of a random variable

$$E[(X-\mu)^2] = \int (x-\mu)^2 p_X(x)dx. \tag{2.34}$$

The variance indicates the concentration of $X$ around its mean. If the variance is small, $X$ is close to $\mu$ with high probability. The variance of a Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ is $\sigma^2$, and we will use $\sigma^2$ to denote the variance of random variables regardless of distribution.

### 2.1.4 Joint Random Variables and Conditioning

Detection and estimation is the study of inferring information about one random variable from another. To do this, we need to describe the statistical relationships between multiple random variables. For continuous random variables $X$ and $Y$, define the **joint probability density function** $p_{XY}(x,y)$. It is non-negative, and it is normalized over *double* integrals:

$$\int\int p_{XY}(x,y)dydx = 1 \tag{2.35}$$

For multiple random variables, we can compute the probability of *joint* events from the PDF. Let $E_1$ and $E_2$ be subsets of $\Gamma_1$ and $\Gamma_2$. Then, the probability that $X \in E_1$ and $Y \in E_2$ simultaneously is

$$\Pr(E_1 \times E_2) = \int_{E_1}\int_{E_2} p_{XY}(x,y)dydx. \tag{2.36}$$

Similarly, for discrete random variables $X \in \Gamma_1$ and $Y \in \Gamma_2$, the **joint probability mass function** $p_{XY}(x,y)$ is just the probability that $X = x$ and $Y = y$ simultaneously. The probability of a joint event $E_1 \subset \Gamma_1$ and $E_2 \subset \Gamma_2$ is the double sum

$$\Pr(E_1 \times E_2) = \sum_{x\in E_1}\sum_{x\in E_2} p_{XY}(x,y). \tag{2.37}$$

A simple example is the flipping of two independent fair coins. Here $\Gamma_1 = \Gamma_2 = \{h,t\}$, and the joint PMF is

$$p(x,y) = 1/4, (x,y) \in \{h,t\} \times \{h,t\}. \tag{2.38}$$

What is the probability of both coins being the same? Of them both being different?

A more complicated example is when $X$ and $Y$ are subsequent letters in a string of English text. The probability of $Y$ ought to depend on the value of $X$: The letter ‘u’ is much more probable if the previous letter is ‘q’. Describing the probabilities of all $26^2$ letter pairs would be too much for an in-class exercise. Instead, the following is the relative probability[3] of each of the two-letter sequences $\{a,b,c,d\} \times \{a,b,c,d\}$:

$$p(x,y) = \begin{bmatrix} 0.0014 & 0.0684 & .2522 & .0708 \\ 0.1078 & 0.0516 & 0.0005 & .0014 \\ 0.21 & 0.0009 & 0.0389 & 0.0014 \\ 0.1725 & 0.0009 & 0.0009 & 0.0202 \end{bmatrix}, \tag{2.39}$$

where the column indicates the value of $X$, and the row indicates the value of $Y$. You can verify that the joint PMF sums to one. The joint PMF tells us something of the *structure* of $X$ and

[3] Collected from `http://norvig.com/mayzner.html`.

$Y$—the way that the values of $X$ and $Y$ are related probabilistically. For example, it's clear that $Y = c$ is most likely if $X = a$, but $Y = a$ is most likely if $X = c$. The structure of the English language means that $X$ and $Y$ tell us something about each other.

From the joint distribution, we can compute the *marginal* distribution, or just the distribution of $X$ and $Y$ individually. We find the marginal distribution by integrating or summing over the joint distribution:

$$p_X(x) = \int p_{XY}(x, y)dy \text{ (continuous)} \tag{2.40}$$

$$p_X(x) = \sum_{y \in \Gamma_2} p_{XY}(x, y)dy \text{ (discrete)}. \tag{2.41}$$

The marginal distribution of the two independent coins is $p(x) = 1/2$ and $p(y) = 1/2$, whereas the marginal distribution of the sequential letters $X$ and $Y$ is more complicated.

#### 2.1.4.1 Independence and Conditioning

The random variables $X$ and $Y$ are **independent** if their PMF or PDF factors, i.e.

$$P_{XY}(x, y) = p_X(x)p_Y(y). \tag{2.42}$$

Independent random variables are mutally non-informative: we cannot learn anything from $X$ by observing $Y$ or vice versa. This property holds for the independent coin tosses: $p(x, y) = 1/4 = p(x)p(y)$. This property does *not* hold for the sequential English letters. We could establish this fact manually by computing $p(x)$ and $p(y)$ from $p(x, y)$ and seeing that $p(x, y) \neq p(x)p(y)$, but we can also intuit this fact. $X$ and $Y$ are dependent because English text has structure.

We can also compute the **conditional** PDF or PMF of $X$ *given* $Y$. That is, suppose we know the value of $Y$; what is the probability of $X$ given this knowledge? The conditional distribution of $X$ given $Y$ is defined as

$$p_{X|Y}(x|y) = \frac{p_{XY}(x, y)}{p_Y(y)}, \tag{2.43}$$

for both continuous and discrete random variables. The conditional distribution gives the (relative) probability that $X = x$ given that $Y = y$. A consequence of this definition is that if $X$ and $Y$ are independent, $p(x|y) = p(x)$. Equivalently, if $X$ and $Y$ are independent, knowing $Y$ doesn't change the probabilities of $X$.

This definition allows us to express the **chain rule** of probability. Rearranging the definition of the conditional PDF and PMF, we get

$$p_{XY}(x, y) = p_{X|Y}(x|y)p_Y(y) = p_{Y|X}(y|x)p_X(x). \tag{2.44}$$

This is called the **chain rule** of probability. Again notice that if $X$ and $Y$ are independent, $p(x|y) = p(x)$.

#### 2.1.4.2 Bayes' Rule

Often, we will have access to $p(x|y)$ and $p(y)$ and want to compute $p(y|x)$. This is possible via **Bayes' rule**:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{2.45}$$

Bayes rule is simple to write down, but it is often difficult to carry out in practice. If we do not know $p(x)$ directly, we have to compute it via the chain rule and marginalization, which for continuous random variables is

$$p(x) = \int p(x, y)dy = \int p(x|y)p(y)dy. \tag{2.46}$$

This integral is hard to carry out in general. Later on we'll see that in a few cases, including Gaussian distributions, we can find $p(y|x)$ without too much trouble. Finding it in more complicated situations has been an active area in Bayesian statistics for several decades.

#### 2.1.4.3 Joint Expectations and Correlation

For joint random variables we often want to look at their joint expectations. By the fundamental theorem of expectation, we can compute the **product expectation**

$$E[XY] = \int\int xyp(x, y)dxdy. \tag{2.47}$$

If $X$ and $Y$ are independent, the expectation factors: $E[XY] = E[X]E[Y]$. This property is particularly useful when either of the random variables has zero mean.

An important use of this is to define the **covariance** between $X$ and $Y$:

$$\rho(X, Y) = E[(X - E[X])(Y - E[Y])]. \tag{2.48}$$

The covariance quantifies how well $X$ describes $Y$. If $X$ and $Y$ are independent, then $\rho(X, Y) = 0$ and the random variables do not describe each other at all. At the other extreme, if $X = Y$, then $\rho(X, Y)$ is equal to the variance of $X$. In this case, we can learn $X$ entirely from $Y$ and vice versa.

#### 2.1.4.4 Multivariate Distributions

We can consider relationships beyond more than two random variables. For $n$ random variables $X_1, X_2, \ldots, X_n$, we can write the joint PDF or PMF $p(x_1, x_2, \ldots, x_n)$. The probability of a joint event is defined in the same way, by integrating or summing over the sets for each random variable. We can also generalize the conditional distribution and Bayes rule by considering *groups* of random variables. Thus,

$$p(x_1|x_2, \ldots, x_n) = \frac{p(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)}. \tag{2.49}$$

Similarly, we can "marginalize out" random variables via integration or summation:

$$p(x_1) = \int\int \ldots \int p(x_1, \ldots, x_n)dx_2 \ldots dx_n. \tag{2.50}$$

An important application of long strings of random variables is the **law of large numbers**. Consider $n$ independent random variables $X_1, X_2, \ldots, X_n$ that have the same distribution $p_X(x)$. Define the **sample mean**, which is just their arithmetic average:

$$Z_n = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{2.51}$$

Now we let $n$ go to infinity. If the variance of $X$ is finite, the sample mean converges on the expectation:

$$Z_n \xrightarrow{P} \mu, \tag{2.52}$$

where the convergence is *in probability.* To be more precise,

$$\lim_{n\to\infty} \Pr(|Z_n - \mu| > \epsilon) = 0, \tag{2.53}$$

for any $\epsilon > 0$. In other words, $E[X]$ the average of $X$ in the following sense: the arithmetic average of many samples from its distribution will be close to $E[X]$. This is what the expectation "means."

Finally, the probably most important multivariate distribution is the **multivariate Gaussian distribution**. Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ be a column vector of random variables. We refer to this as a **random vector**. We say that the random vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if the joint distribution is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{2.54}$$

where $\mu$ is a vector that gives the mean of each $X_i$, and $\Sigma$ is the $n \times n$ **covariance matrix**, whose entries are

$$\Sigma_{ij} = \rho(x_i, x_j). \tag{2.55}$$

The diagonal entries of $\boldsymbol{\Sigma}$ are the variances of each $x_i$, and the off-diagonal entries tell us how correlated the random variables are with each other. If $x_i$ and $x_j$ are independent, they have no correlation, and $\Sigma_{ij} = 0$. We can also write the covariance matrix in terms of the random vector. Specifically[4],

$$\boldsymbol{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]. \tag{2.56}$$

One reason Gaussians are so popular is that they "play nice" with linear systems. In particular, we can extend the linearity of the expectation to multivariate distributions expressed as random vectors. Specifically, let $E[\mathbf{x}]$ be the expected value of a random *vector*, i.e. the vector of expectations of $\mathbf{x}$. Then, for matrix $\mathbf{A}$, $E[\mathbf{A}\mathbf{x}] = \mathbf{A}E[\mathbf{x}]$. Using this linearity with the preceding definition of the covariance matrix, we also get a formula for the covariance of $\mathbf{A}\mathbf{x}$:

$$E[(\mathbf{A}\mathbf{x}-\mathbf{A}\boldsymbol{\mu})(\mathbf{A}\mathbf{x}-\mathbf{A}\boldsymbol{\mu})^T] = \mathbf{A}E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]\mathbf{A}^T = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T. \tag{2.57}$$

We will frequently make use of this formula when dealing with Gaussian random vectors.

It turns out that multivariate Gaussians interact particularly well with linear systems, and the linear-algebraic structure of the distribution will give us insight into what is going on. To explore this further, we'll need to review briefly linear algebra.

## 2.2 Review of Linear Algebra

Linear algebra gives us a convenient way to structure our data. In general, a vector $\mathbf{x}$ represents a *signal*, a matrix $\mathbf{A}$ represents a *system*, and the result of applying signal $\mathbf{x}$ to system $\mathbf{A}$ is the output $\mathbf{b} = \mathbf{A}\mathbf{x}$. We will generally use boldface lowercase letters to refer to vectors, and boldface capitals to refer to matrices.

[4] The matrix transpose is defined in the following section.

### 2.2.1 Vectors and Matrices

We will let $\mathbf{x} \in \mathbb{R}^n$ be an $n$-length **column vector**:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.58}$$

We will also denote an $m \times n$ matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}. \tag{2.59}$$

That is, $a_{ij}$ refers to the element of $\mathbf{A}$ in the $i$th row and the $j$th column. The **transpose** of the matrix $\mathbf{A}$ reflects the matrix across the diagonal:

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}. \tag{2.60}$$

A matrix such that $\mathbf{A} = \mathbf{A}^T$ is called **symmetric**. Many important matrices are symmetric, including covariance matrices!

We will let $\mathbf{I}$ denote the **identity matrix**, which has ones on the diagonal and zeros elsewhere. A square matrix $\mathbf{A}$ is **invertible** if there is a matrix $\mathbf{A}^{-1}$ such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, and $\mathbf{A}^{-1}$ is its inverse. Not every square matrix is invertible, and a matrix that is not square cannot have an inverse.

A matrix $\mathbf{A}$ is **diagonal** if it only has nonzero entries on its main diagonal, i.e. for values of $a_{ii}$. Sometimes we will "build" a square, diagonal matrix from a vector $\mathbf{v} \in \mathbb{R}^n$ via the notation $\mathbf{A} = \text{diag}(\mathbf{v})$, which means we place the entries of $\mathbf{v}$ on the main diagonal of $\mathbf{A}$.

### 2.2.2 Norms and Inner Products

The **inner product** (or **dot product**) of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is defined as

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y} = \sum_{i=1}^n x_i y_i. \tag{2.61}$$

The inner product tells us how similar two vectors are in terms of *direction*. Vectors that point in the same direction have high inner products, and vectors that point at right angles to each other have small inner product. Specifically, we say that two vectors are **orthogonal** if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Orthogonality is a geometric way of saying that two vectors have nothing in common. The vectors $\mathbf{x} = (0, 1)^T$ and $\mathbf{y} = (1, 0)^T$ are orthogonal, as are the vectors $\mathbf{x} = (1, 1)$ and $\mathbf{y} = (1, -1)$.

The **norm** of a vector is defined as

$$\|\mathbf{x}\| = (\langle \mathbf{x}, \mathbf{x} \rangle)^{1/2} = \left( \sum_{i=1}^n x_i^2 \right)^{1/2}. \tag{2.62}$$

In geometric terms, the norm tells us the distance of $\mathbf{x}$ to the origin. In signal processing terms, the norm tells us something about the *energy* in the signal $\mathbf{x}$. We'll often use the *squared norm* $\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle$ to measure the signal energy. If $\|\mathbf{x}\| = 1$, we say that $\mathbf{x}$ is **normalized** or is a **unit vector**.

This gives rise to a special class of matrices. We say that a square matrix $\mathbf{U} \in \mathbb{R}^{n \times n}$ is **orthonormal** or **unitary** if its columns are unit vectors and orthogonal with each other, i.e.

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_n \end{bmatrix}, \tag{2.63}$$

where $\|\mathbf{u}_i\| = 1$ and $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$. An orthonormal matrix gives us a basis set for the set of vectors $\mathbb{R}^n$. We can decompose any vector $\mathbf{x}$ in terms of the columns of $\mathbf{U}$. We'll find orthonormal matrices indispensable in dealing with linear systems and signals with a multivariate Gaussian distribution, as we'll see shortly.

A useful property of orthonormal matrices, which you can easily verify, is that $\mathbf{A}^{-1} = \mathbf{A}^T$. An orthonormal matrix is rather easy to invert.

### 2.2.3 Eigenvectors and Eigendecompositions

You probably remember that a vector $\mathbf{v} \in \mathbb{R}^n$ is an **eigenvector** of $\mathbf{A} \in \mathbb{R}^{n \times n}$ (with eigenvalue $\lambda$) if

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}. \tag{2.64}$$

We are primarily interested in eigenvectors in order to decompose $\mathbf{A}$ into simpler pieces. Specifically, we want to describe $\mathbf{A}$ entirely in terms of how it acts on its eigenvectors. If $\mathbf{A}$ has $n$ *linearly independent* eigenvalues, then we can express $\mathbf{A}$ in terms of its **eigendecomposition**:

$$\mathbf{A} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^{-1}, \tag{2.65}$$

where $\mathbf{V}$ is the matrix of eigenvectors, and $\boldsymbol{\lambda}$ is the vector of the eigenvalues corresponding to each eigenvector.

In the special case of $\mathbf{A}$ symmetric, it is possible to show that the eigenvectors are orthogonal to each other. As a result, the eigendecomposition can be written in terms of an orthonormal matrix $\mathbf{V}$:

$$\mathbf{A} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^T. \tag{2.66}$$

An orthonormal eigendecomposition allows us to break up $\mathbf{A}$ into its constituent behaviors. We can look at how it acts on each of its eigenvectors $\mathbf{v}_i$, and then sum up all of those actions in order to find out how $\mathbf{A}$ acts on an arbitrary vector $\mathbf{x}$.

Recall that the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ of a multivariate Gaussian distribution is symmetric. Therefore it has an orthonormal eigendecomposition:

$$\boldsymbol{\Sigma} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^T. \tag{2.67}$$

The eigendecomposition reveals the geometric structure of a multivariate Gaussian. If $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, then we can think of $\mathbf{x}$ as the sum of $n$ independent random vectors, each pointing in the direction of the eigenvectors $\mathbf{v}_i$ and having variance $\lambda_i$. To see this, let's consider the random vector $\mathbf{w} \sim \mathcal{N}(0, \text{diag}(\boldsymbol{\lambda}))$. This is a vector of independent random variables with variances $\lambda_i$. Next, we multiply $\mathbf{w}$ by the matrix of eigenvectors:

$$\mathbf{z} = \mathbf{V}\mathbf{w} = \sum_{i=1}^{n} \mathbf{v}_i w_i. \tag{2.68}$$

From the discussion in the previous section, we know that $\mathbf{z} \sim \mathcal{N}(0, \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^T)$, which is the distribution of the original signal $\mathbf{x}$. As we claimed, $\mathbf{x}$ is simply a linear combination of independent Gaussian random variables, with variances equal to the eigenvectors of $\boldsymbol{\Sigma}$ and multiplied by the eigenvectors of $\boldsymbol{\Sigma}$.

Conversely, we can take $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ and obtain a "white," or uncorrelated vector of Gaussians. Define the operator $\mathbf{A} = (\sqrt{\text{diag}(\boldsymbol{\lambda})})^{-1}\mathbf{V}^T$, where the square root is applied to each element of the matrix. Then, defining $\mathbf{u} = \mathbf{A}\mathbf{x}$, we obtain a Gaussian with identity covariance:

$$\mathbf{u} \sim \mathcal{N}(0, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T) = \mathcal{N}(0, (\sqrt{\text{diag}(\boldsymbol{\lambda})})^{-1}\mathbf{V}^T\mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^T\mathbf{V}\sqrt{\text{diag}(\boldsymbol{\lambda})})^{-1}) = \mathcal{N}(0, \mathbf{I}), \tag{2.69}$$

where the simplification is due to the fact that $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. The matrix $\mathbf{A} = (\sqrt{\text{diag}(\boldsymbol{\lambda})})^{-1}\mathbf{V}^T$ is often called a **whitening filter**. It allows us to transform a Gaussian vector with arbitrary correlations into a vector that is uncorrelated.

## 2.3 Introduction to Optimization Theory

Often we will solve a detection or estimation problem by finding the solution that gives the *minimum* error—that is, the detector/classifier that has the smallest probability of error, or the estimator of an image whose output is closest to the original image. Such problems are cast as *optimization* problems, in which we *minimize* a function subject to some constraints.

We formalize optimization by defining a function (sometimes called the **cost function**) $f(\mathbf{x})$, which takes a vector $\mathbf{x} \in \mathbb{R}^n$ and outputs a scalar. The cost function might represent the average error associated with choosing $\mathbf{x}$ as an estimate, or the average error in fitting a statistical model $\mathbf{x}$ to a set of data. There are many examples outside of data science, too; optimization is ubiquitous throughout engineering, physics, economics, and the quantitative sciences in general.

We write down the problem of minimizing $f(\mathbf{x})$ as

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{2.70}$$

and we write down the *solution* to the problem as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}). \tag{2.71}$$

In other words, $\min_{\mathbf{x}} f(\mathbf{x})$ is the smallest value of $f(\mathbf{x})$ for any input, and $\arg\min_{\mathbf{x}} f(\mathbf{x})$ is the $\mathbf{x}$ that *achieves* this smallest value. We usually let $\mathbf{x}^*$ denote the input that optimizes $f(\mathbf{x})$. As a very simple example, let $f(x) = (x-2)^2 + 1$. This quadratic function is minimized when $x = 2$, and its minimum value is $f(2) = 1$; thus $\min_x f(x) = 1$ and $x^* = \arg\min_x f(x) = 2$.

Optimization is easiest to carry out when $f(\mathbf{x})$ is a **convex** function, defined as a function with the property

$$f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \le \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2), \tag{2.72}$$

for any two points $\mathbf{x}_1, \mathbf{x}_2$ and for $0 \leq \alpha \leq 1$. Visually, a convex function is one that is "concave up" as depicted in Figure 2.1. By contrast, a **concave** function is one such that $f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \geq \alpha f(\mathbf{x}_1) + (1-\alpha)f(\mathbf{x}_2)$, or equivalently $-f(\mathbf{x})$ is convex. Many functions are neither concave nor convex.

Elementary calculus tells us that if we want to find a maximum or minimum of a function, we should set its derivative to zero. Since $f(\mathbf{x})$ is a multivariate function, we instead need to work with its **gradient**, which is the vector defined as

$$\nabla_{\mathbf{x}} f(\mathbf{x}) := \left( \frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)^T. \tag{2.73}$$

An extremely common starting place for optimization is to set $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$, solve for $\mathbf{x}$, and call the result the solution $\mathbf{x}^*$. If the gradient exists and $f(\mathbf{x})$ is convex, any point $\mathbf{x}$ such that $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$ is guaranteed to be a **global minimum** of $f(\mathbf{x})$, meaning that there is no other point $\mathbf{y}$ such that $f(\mathbf{y}) \leq f(\mathbf{x})$. If $f(\mathbf{x})$ is *not* convex, then the point $\mathbf{x}$ is called a **stationary point**. Stationary points may be global minima, but they may also be **local minima**, meaning that they minimize $f(\mathbf{x})$ in their immediate neighborhood, **local or global maxima**, meaning that they *maximize* $f(\mathbf{x})$, or **saddle points**, meaning that they are local maxima along one dimension of $\mathbf{x}$ but local minima along another. In Figure 2.1, setting the gradient equal to zero gives a global minimum for the first (convex) function, a global maximum for the second (concave) function, and a both local maxima and minima for the third (neither convex nor concave) function.

Many of the problems we will work with are convex, in which case local optimality and saddle points are not an issue. However, we will frequently find that it is not possible to solve $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$ directly.

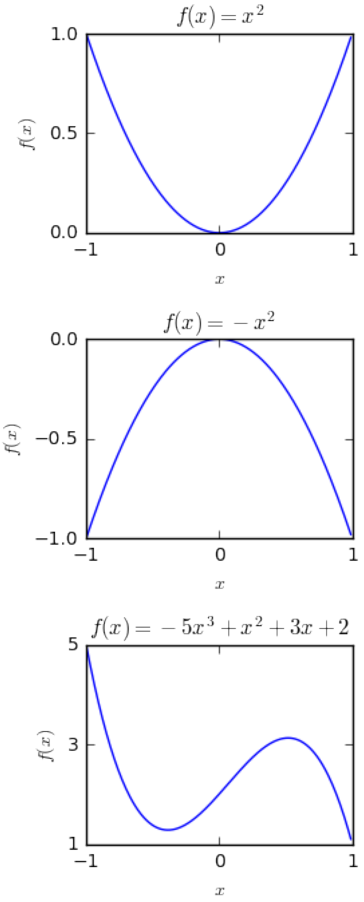

Figure 2.1: Functions that are convex, concave, and neither, respectively.

### 2.3.1 Iterative Optimization: Gradient Descent

When it is difficult (or impossbile) to solve $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$ directly, we can appeal to *iterative* optimizaiton methods, by far the most famous and often-used of which is **gradient descent**. In gradient descent, we move gradually towards the solution $\mathbf{x}^*$ by following the gradient, which points in the direction of *steepest descent*. We formalize this by defining a sequence of points $\{\mathbf{x}_k\}_{k=0}^{\infty}$. We initialize the sequence with $\mathbf{x}_0$ arbitrarilty, and we update the elements of the sequence by stepping in the direction of the gradient:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{2.74}$$

where $\alpha_k$ is the **stepsize**, also called the **learning rate**. In other words, at each iteration $k$, we compute the gradient of $f(\mathbf{x})$ evaluated at the point $\mathbf{x}_k$, take a small step in the direction of the gradient, and continue.

The choice of step size induces a trade-off: Larger $\alpha_k$ speeds up convergence towards $\mathbf{x}^*$, but if $\alpha_k$ is too large the sequence may not converge at all! A variety of methods exist for choosing $\alpha_k$, and they have varying complexity and performance guarantees. We'll mostly ignore them in this course, but you should know to go looking for them if you are using gradient descent in your research. Instead, we'll focus on the most popular choice in practice: **constant** stepsize (i.e. $\alpha_k = \alpha$).

Another issue is the choice of *stopping conditions*. The most principled choice is to continue iterating until the gradient is small, i.e. $\|\nabla f(\mathbf{x}_k)\| < \epsilon$ for some small threshold. This ensures that the final iterate $\mathbf{x}_k$ approximately solves $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$, which is the point of this exercise!

When $f(\mathbf{x})$ is convex, and $\nabla f(\mathbf{x})$ satisfies a technical condition called *Lipschitz continuity*, gradient descent is guaranteed to converge on the global minimum $\mathbf{x}^*$. If $f(\mathbf{x})$ is not convex, gradient descent converges on a stationary point. Even these stationary points will often perform well enough in practice, and as a result gradient descent is extremely common for solving non-convex problems.

In the grand scheme of optimization schemes, gradient descent actually has rather slow convergence. By contrast, methods stuch as the Newton-Raphson method take advantage of higher-order moments to find more quickly the minimizer (or a stationary point). However, gradient descent is simple and computationally cheap, and as a result is a workhorse in modern optimization.

Research on iterative optimization methods is a very active currently, much of it driven explicitly by problems in machine learning.

### 2.3.2 Constrained Optimization: The Karush-Kuhn-Tucker Conditions

Often we will carry out **constrained optimization**, in which we want to minimize the cost function $f(\mathbf{x})$ while imposing conditions on the solution $\mathbf{x}^*$. For example, we may want to constrain the energy of the solution by fixing its norm. We encode the constraints via functions $g_i(\mathbf{x})$, which define **equality constraints** of the form $g_i(\mathbf{x}) = 0$, and functions $h_i(\mathbf{x})$, which define **inequality constraints** of the form $h_i(\mathbf{x}) \leq 0$. Together, these functions define the constraints that determine the optimization problem to be solved. We can write this problem formally as

$$\underset{\mathbf{x}, g_i(\mathbf{x})=0, h_i(\mathbf{x})\leq 0}{\arg\min} \quad f(\mathbf{x}), \tag{2.75}$$

and we still denote the constrained minimizer by $\mathbf{x}^*$. We will suppose that $f(\mathbf{x})$, $h_i(\mathbf{x})$, and $g_i(\mathbf{x})$ are all convex functions.

We can solve this problem using an extention of the **method of Lagrange multipliers**. This method involves first defining the **Lagrangian function**:

$$L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x}) + \sum_j \mu_j h_j(\mathbf{x}). \tag{2.76}$$

That is, we augment the function $f(\mathbf{x})$ to be minimized by the constraints. The terms $\lambda_i$ and $\mu_i$ are Lagrange multipliers associated with each constraint. Then, the **Karush-Kuhn-Tucker (KKT)**

**conditions**[5] state the the constrained minimizer $\mathbf{x}^*$ satisfies the following conditions:

$$\begin{aligned} \nabla_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu) &= 0 && (2.77) \\ \nabla_{\lambda} L(\mathbf{x}, \lambda, \mu) &= 0 && (2.78) \\ \mu &\geq 0 && (2.79) \\ \mu_j h_j(\mathbf{x}) &= 0. && (2.80) \end{aligned}$$

The first condition means that the optimizer $\mathbf{x}^*$ is a stationary point of the Lagrangian instead of merely of $f(\mathbf{x})$. The second condition implicity insists that $\mathbf{x}^*$ satisfy the equality constraints. (Try taking the derivative with respect to $\lambda_i$ to see why.) The third and fourth conditions make sure that the inequality constraints are satisfied by enforcing a condition called **complementary slackness**. The intuition goes like this: If an inequality constraint $h_j(\mathbf{x})$ holds with equality, then we call it an **active constraint**, and its Lagrange multiplier $\mu_j$ is positive to make sure that the Lagrangian incorporates the constraint. But if $h_j(\mathbf{x})$ does *not* hold with equality, then we treat the constraint as if it doesn't exist, letting $\mu_j = 0$ to "zero out" the constraint's contribution to the Lagrangian. Note that we do not know in advance which constraints will be active and which will not.

The KKT conditions lead to a fairly simple algorithm for solving constrained, convex problems:

- Differentiate $L(\mathbf{x}, \lambda, \mu)$ with respect to $\mathbf{x}$ and solve for $\mathbf{x}^*$ in terms of the (as yet unknown) Lagrange multipliers $\lambda_i$ and $\mu_j$.
- Find the values of $\lambda_i$ that ensure that the equality constraints are met.
- Try out all possible active/inactive combinations on the inequality constraints, solving for $\mu_j \geq 0$ that ensure that the active constraints are met with equality.

The final step of this procedure is the trickiest. If there are many inequality constraints, it may be time-consuming to iterate over all of the active/inactive combinations. Nevertheless, for relatively small or simple problems, the KKT conditions provide a powerful framework for solving constrained problems, and they are ubiquitous in the signal processing and machine learning literature.

For larger problems, or problems where we cannot carry out these steps directly, one can adopt iterative methods to incorporate constraints. The most popular of these is **projected gradient descent**, in which the usual gradient step is followed by "projecting" the iterate to the nearest point that satisfies the constraints.

[5] For a long time, these conditions were called the KT or Kuhn-Tucker conditions, after the mathematicians who historically have been best known for this method. It was later discovered that Karush anticipated Kuhn and Tucker's result by more than a decade; since then it has become steadily more common to speak of the KKT conditions rather than the KT conditions.

# Chapter 3

# Detection Theory: Hypothesis Testing

This chapter covers the detection problem, which in the language of statistics we frame in terms of **hypothesis testing**. We can state the problem formally as follows. Let the variable $H \in \{0, 1, \ldots, M-1\}$ represent one of $M$ possible "states of nature". These states could represent the presence (and/or type) of object present in a radar return or an image, or which of $M$ digital codewords were sent over a wireless channel. We observe a signal or random variable $Y$ whose distribution depends on which state of nature $H$ is true. The observation $Y$ may be a discrete or continuous random variable or random vector; in the case of a random vector we will let $\mathbf{y}$ denote the observed signal.

The detection problem is to infer $H$ from the observed signal $Y$. We express the relationship between $H$ and $Y$ by defining the **likelihood function** for $Y$ for each of the $M$ hypotheses:

$$\begin{aligned} H_0 &: Y \sim p(y|H=0) \\ H_1 &: Y \sim p(y|H=1) \\ &\vdots \\ H_{M-1} &: Y \sim p(y|H=M-1) \end{aligned}$$

where each $p(y|H=i)$ is a density or mass function as appropriate. That is, each hypothesis $H_i$ is true if $H=i$, and for each hypothesis there is a different distribution on the signal $Y$.

We need to take a bit of care with the likelihood function, as it bumps up against a philosophical issue. There are two dominant approaches to detection[1]: **Bayesian** and **Neyman-Pearson**. In the Bayesian approach, we let $H$ be a random variable, with its own probability distribution, and we speak freely of the probability that $H$ takes on a particular value. In this setting, the likelihood $p(y|h)$ is a conditional distribution, and we use Bayes rule to compute the distribution $p(h|y)$ and solve the detection problem.

By contrast, in the Neyman-Pearson approach we assume that $H$ is *not* a random variable. Instead, $H$ simply *is* whatever value it is, and it makes no sense to assign it a probability distribution. In this case, the likelihood function $p(y|H=i)$ is *not* a conditional distribution[2], but merely one of $M$

[1] A similar divide is present in estimation, too, as we will see in the next chapter.

[2] In fact, many authors are careful to use notation like $p(y;H)$ in the Neyman-Pearson setting to avoid suggesting that the likelihood function is a conditional distribution. Because I am generally Bayesian in my philosophical outlook, I find this to be more trouble than it is worth.

possible distributions for $Y$. We speak of probabilities for $Y$ *supposing* that one of the hypotheses is true, but never vice versa.

## 3.1 Binary Hypothesis Testing

We will start with the simplest possible detection problem: **binary hypothesis testing**. The are only two hypotheses, $H_0$ and $H_1$:

$$\begin{aligned} H_0 &: Y \sim p(y|H=0) \\ H_1 &: Y \sim p(y|H=1), \end{aligned}$$

If $H_0$ is true, then the distribution of $Y$ is $p(y|H=0)$, and if $H_1$ is true, the distribution of $Y$ is $p(y|H=1)$. It is common to refer to $H_0$ as the **null hypothesis** and $H_1$ as the **alternative hypothesis**. Roughly speaking, the null hypothesis represents the expected or "default" state of nature, whereas the alternative hypothesis represents an unusual or "interesting" state. In radar detection, for example, the observation $Y$ might be the amplitude of the radar return signal, and the null and alternative hypotheses may correspond to the presence and absence of an object. This terminology is common in science, where researchers design experiments to test the relationship between phenomena. The null hypothesis corresponds to no relationship—a drug does not cure a disease, the consumption of bacon has no impact on the incidence of cancer[3], etc. If the experiment indicates a relationship, we often say that the null hypothesis is *rejected*.

Let's look at a few examples of binary hypothesis testing.

**Example 3.1: Simple Object Detection.** *This is an extremely simplified version of object detection with radar. Suppose we emit a pulsed radar signal in search of a potential object—an aircraft, etc. We take the null hypothesis $H_0$ to be the case in which there is no object, in which case the radar pulse does not return and the receiver signal is only noise. We take the alternative hypothesis $H_1$ to be the case in which there* is *an object, in which case the pulse reflects and returns, and the receiver signal is the reflected pulse plus noise. We model these scenarios by simple, univariate Gaussian likelihood functions:*

$$p(y|H=0) = \mathcal{N}(0, \sigma^2) \tag{3.1}$$

$$p(y|H=1) = \mathcal{N}(a, \sigma^2), \tag{3.2}$$

*where $\sigma^2$ represents the noise power, and $a$ represents the amplitude of the returned radar pulse. The larger the amplitude $a$, and the smaller the noise power $\sigma^2$, the easier is the detection problem.*

**Example 3.2: Spam Detection.** *A more realistic example is the construction of an email spam filter. The null hypothesis $H_0$ is that the email is an ordinary, non-spam message, and the alternative hypothesis $H_1$ is that the email is spam. A simple but powerful approach to spam filtering—and to text processing in general—is the* **bag of words** *model, in which we examine the frequency of selected words in the document, but we ignore their position. In spam detection, the received signal is a vector $\mathbf{y} \in \mathbb{Z}_+^k$ of $k$ integers, where $k$ is the number of vocabulary words considered, and each $Z_i$ gives the number of times the $i$th word appears in the email message we are trying to classify.*

*We construct the likelihood functions by supposing that under the* null hypothesis *each word appears with probabilities $p_1^0, p_2^0, \ldots, p_k^0$, and under the* alternative hypothesis *each word appears*

[3] https://www.wired.com/2015/10/who-does-bacon-cause-cancer-sort-of-but-not-really/

*with probabilities* $p_1^1, p_2^1, \cdots, p_k^1$. *These probabilities are different because spam messages tend to use certain words—like "sale", "watch", or "wire"—more often than in ordinary writing. If we assume that a document of* $n$ *total words has words drawn i.i.d. according to these distributions, than the vector* $\mathbf{y}$ *of word counts has a* **multinomial distribution**, *which gives us the following likelihood functions:*

$$p(y|H=0) = \frac{n!}{y_1! \cdots y_k!}(p_1^0)^{y_1} \cdot (p_2^0)^{y_2} \cdots (p_k^0)^{y_k} \tag{3.3}$$

$$p(y|H=1) = \frac{n!}{y_1! \cdots y_k!}(p_1^1)^{y_1} \cdot (p_2^1)^{y_2} \cdots (p_k^1)^{y_k}, \tag{3.4}$$

*where the factorial terms ensure normalization, and where we assume that* $\sum_k y_k = n$ *since the document has* $n$ *total words. Note that these likelihood functions are PMFs, since the signal is discrete. Here, the more distinct the probabilities* $p_i^0, p_i^1$, *the easier it will be to tell a spam message apart from an ordinary one.*

The main work of this section will be the design of optimal detectors. Given likelihood functions $p(y|H=0)$ and $p(y|H=1)$ and an observation $Y$, how do we choose best between $H_0$ and $H_1$? The answer is sensitive to how we define "best", and it turns out that it is not always straightforward to do so. Consider radar detection. Which is worse: to miss detecting an object—that is, to decide that $H_0$ is true when in fact $H_1$ is true—or to sound a false alarm—to decide that $H_1$ is true when in fact $H_0$ is true? The answer is context-dependent at best and completely arbitrary at worst.

This chapter will describe the three most common formulations to the question: **Neyman-Pearson hypothesis testing**, **Bayesian hypothesis testing**, and **minimax hypothesis testing**. None of these approaches settles the questions conclusively, but each provides a way to think systematically about the issues at play.

### 3.1.1 Neyman-Pearson Hypothesis Testing

We begin with the most widely-known approach to hypothesis testing, the **Neyman-Pearson test**. Indeed, Neyman-Pearson theory is prior to the Bayesian and minimax frameworks, and notions that are ubiquitous in experimental science—null and alternative hypotheses, p-values, type I and type II errors, etc.—are grounded in Neyman-Pearson theory.

First, we define a detector in the abstract. Let $\Gamma$ be the sample space for $Y$, i.e. the set of values it can take on—whether the real line, a vector space, or a discrete set. The role of the detector is to input the received signal $Y$ and output a decision of whether it thinks the null hypothesis ($H_0$) or the alternative hypothesis ($H_1$) is true.

Formally, then, all a detector does is partition the set $\Gamma$ into two regions: signal values $Y$ for which it decides $H_0$ is true, and values $Y$ for which it decides that $H_1$ is true. This leads to the following formal definition of a detector.

**Definition 3.1:.** *For sample space* $\Gamma$, *let the sets* $\Gamma_0$ *and* $\Gamma_1$ *be a partition of* $\Gamma$, *i.e.* $\Gamma_0, \Gamma_1 \subset \mathcal{G}$ *and* $\Gamma_0 = \Gamma_1^c$. *Then, a* **decision rule** $\delta$ *is the function*

$$\delta(y) = \begin{cases} 0 & \textit{if } y \in \Gamma_0 \\ 1 & \textit{if } y \in \Gamma_1 \end{cases}. \tag{3.5}$$

The detector $\delta(y)$ can be any binary partition of the sample space, but we want to choose it in an optimal fashion. But optimal with respect to what?

In Neyman-Pearson theory the roles of $H_0$ as the null hypothesis and $H_1$ as the alternative hypothesis are crucial. The null hypothesis is the "boring" hypothesis, representing the status quo, whereas the alternative hypothesis is the "interesting" one, suggesting that something unusual or anomalous is afoot. Each hypothesis has associated with it a different *type* of error. We can reject the null hypothesis when in fact $H_0$ is true, or we can fail to reject the null hypothesis when $H_1$ is true. In Neyman-Pearson theory, we call the first error a **Type I** error, or a "false alarm". We call the second error a **Type II** error or a "missed detection". We want to minimize both of these error types, but these desires are in conflict. A conservative detector may declare few false alarms, but it will also miss many detections!

Instead, the goal of the Neyman-Pearson detector is to give an optimum trade-off between these two types of error. To do so, we formalize these notions.

**Definition 3.2:.** *For a binary hypothesis test with detector $\delta(y)$, the* **false-alarm probability**, *also called the* **size** *of the test, is*

$$P_F(\delta) = \Pr(\delta(Y) = 1 | H = 0), \tag{3.6}$$

*the* **missed detection probability** *is*

$$P_M(\delta) = \Pr(\delta(Y) = 0 | H = 1), \tag{3.7}$$

*and the* **detection probability**, *also called the* **power** *of the test, is*

$$P_D(\delta) = 1 - P_M(\delta) = \Pr(\delta(Y) = 1 | H = 1). \tag{3.8}$$

The premise of Neyman-Pearson theory is to fix the size[4] of the test and find the most powerful detector. That is, the design criterion is

$$\delta^* = \arg\max_{\delta} P_D(\delta), \text{ subject to } P_F(\delta) \leq \alpha, \tag{3.9}$$

where $\alpha$ is chosen in advance. Neyman-Pearson theory mediates the trade-off between power and size by fixing the size in advance. We choose to tolerate a certain probability of false alarm, and we find the detector that maximizes the probability of successful detection. We can always find one that satisfies a *different* trade-off, by choosing a different $\alpha$, but no detector can achieve a better trade-off than the one satisfying the Neyman-Pearson criterion.

So far we have only defined the *criterion* for the Neyman-Pearson test, not the test itself. In fact, the Neyman-Pearson criterion expressed in (3.9) is a constrained optimization problem! But rather than build a Lagrangian and invoke the KKT conditions, we'll make a direct argument that shows that the optimum detector has an important form, which we will see often: the **likelihood ratio test**, often abbreviated **LRT**. That is, we compute a quantity called the **likelihood ratio**:

$$l(y) := \frac{p(y|H=1)}{p(y|H=0)}, \tag{3.10}$$

[4] In the machine learning literature, the terms *size* and *power* are used less frequently in favor of descriptions of the **precision** and **recall** of a classifier. Recall measures the fraction of $H_1$ events that are detected, which is essentially

which we compare to a threshold to be determined later. If $l(y)$ is larger than this threshold, we let $\delta(y) = 1$. The threshold of the LRT is determined by the size $\alpha$ of the test. The larger the size $\alpha$, the more tolerant we are to Type I errors, and the lower the threshold on the likelihood ratio.

**Lemma 3.1: Neyman-Pearson Lemma.** *Let $\delta(y)$ be a randomized likelihood ratio test of the form*

$$\delta(y) = \begin{cases} 0 & \textit{if } l(y) < \eta \\ \gamma(y) & \textit{if } l(y) = \eta \ , \\ 1 & \textit{if } l(y) > \eta \end{cases} \tag{3.11}$$

*where the second condition means that the detector returns a* 1 *with probability $\gamma(y)$ and a* 0 *with probability $1 - \gamma(y)$. Then three statements hold:*

*(i) $\delta(y)$ is the optimum test for its size. Let $\alpha = P_F(\delta)$ be the size of the test. No other detector $\delta'$ with size $\alpha$ has $P_D(\delta') > P_D(\delta)$.*

*(ii) For every $\alpha$, there exists a threshold $\eta$ and a constant $\gamma(y) = \gamma_0$ such that $P_F(\delta) = \alpha$.*

*(iii) Any detector $\delta'$ of size $\alpha$ that has $P_F(\delta') = P_F(\delta)$ has the form (3.11) except over a set having probability zero under $H_0$ and $H_1$.*

Before proving the lemma, let's expand on the role of the *randomized* LRT. In the event that the likelihood ratio is exactly equal to $\eta$, we in essence flip a weighted coin to break the tie. This can only effect the size and power of the test in circumstances where a tie occurs with nonzero probability. For continuous detection problems, this is rarely the case: the likelihood ratio is usually a continuous random variable, and the probability that it equals any specific value is zero. In cases where ties do happen with nonzero probability, randomization is a way to ensure that test of all sizes $\alpha$ are possible. Otherwise, the size of the test will be discontinuous with the LRT ratio $\eta$, and some values of $\alpha$ will not be achievable. In practice, the inclusion of a randomized rule doesn't change much. It allows us formally to say that an LRT is optimum for every $\alpha$, but it doesn't result in a better trade-off between Type I and Type II errors.

*Proof of the Neyman-Pearson Lemma.* To prove (i), we need to show that $P_D(\delta) \geq P_D(\delta')$ for any test $\delta'$ of size no greater than $\alpha$. Consider the following chain of inequalities:

$$\begin{aligned} P_D(\delta) - P_D(\delta') &= P_D(\delta) - P_D(\delta') + \eta[P_F(\delta) - P_F(\delta')] - \eta[P_F(\delta) - P_F(\delta')] && (3.12) \\ &= P_D(\delta) - P_D(\delta') + \eta[\alpha - P_F(\delta')] - \eta[P_F(\delta) - P_F(\delta')] && (3.13) \\ &\geq P_D(\delta) - P_D(\delta') - \eta[P_F(\delta) - P_F(\delta')], && (3.14) \end{aligned}$$

because $P_F(\delta') \leq \alpha$ by hypothesis. Continuing, we have

$$\begin{aligned} P_D(\delta) - P_D(\delta') &\geq \int_\Gamma (\delta(y) - \delta'(y))p(y|H=1)dy - \eta \int_\Gamma (\delta(y) - \delta'(y))p(y|H=0) && (3.15) \\ &= \int_\Gamma [\delta(y) - \delta'(y)][p(y|H=1) - \eta p(y|H=0)]dy. && (3.16) \end{aligned}$$

Consider the integrand $[\delta(y) - \delta'(y)][p(y|H=1) - \eta p(y|H=0)]$. When $p(y|H=1) - \eta p(y|H=0)$ is positive, $\delta(y) = 1$, so the product is positive. Similarly, when $p(y|H=1) - \eta p(y|H=0)$ is

equivalent to the power of the test. Precision measures the ratio of detections that are correct, which corresponds to (but is not equal to!) the size of the test.

negative, $\delta(y) = 0$, so the product is positive. When $p(y|H = 1) - \eta p(y|H = 0) = 0$ the product is of course zero. Therefore $[\delta(y) - \delta'(y)][p(y|H = 1) - \eta p(y|H = 0)] \geq 0$, thus

$$P_D(\delta) - P_D(\delta') \geq \int_\Gamma [\delta(y) - \delta'(y)][p(y|H = 1) - \eta p(y|H = 0)]dy \geq 0. \tag{3.17}$$

(ii) is trivial if the likelihood ratio has a continuous distribution under $H_0$; we can sweep $\alpha$ from zero to one by letting the ratio $\eta$ vary. Otherwise, we can choose $\gamma_0$ to achieve any $\alpha$ in the regions where the size of the test is discontinuous in $\eta$.

To prove (iii), first observe that for any $\delta'$ that satisfies the Neyman-Pearson criterion, $P_D(\delta') = P_D(\delta)$ and $P_F(\delta) = P_F(\delta')$. By the previous rearrangement of terms,

$$P_D(\delta) - P_D(\delta') - [P_F(\delta) - P_F(\delta')] = \int_\Gamma [\delta(y) - \delta'(y)][p(y|H = 1) - \eta p(y|H = 0)]dy = 0. \tag{3.18}$$

As argued before, the integrand is positive everywhere. Therefore, the integral is zero only if $\delta(y) - \delta'(y)$ is zero everywhere except for a set of probability zero, and possibly over the tiebreaker set for which $p(y|H = 1) - \eta p(y|H = 0)$. Therefore it has the Neyman-Pearson form almost everywhere. □

Now that we've proven the optimality of the LRT—at least, with respect to the Neyman-Pearson criterion—let's try it out on a few examples. Because so many distributions involve exponential functions, we will often express an LRT in terms of the **log-likelihood ratio** (LLR), defined as

$$L(y) := \log l(y) = \log p(y|H = 1) - \log p(y|H = 0), \tag{3.19}$$

where we use the natural log. Because the logarithm is a monotonically increasing function, the LRT $l(y) \lessgtr \eta$ can be expressed as a test on the LLR, or $L(y) \lessgtr \nu$, where $\nu := \log \eta$.

**Example 3.3: Simple Object Detection.** *Let's revisit the simple detection system in which $H_0$ and $H_1$ have the following likelihood functions:*

$$\begin{aligned} p(y|H = 0) &= \mathcal{N}(0, \sigma^2) \\ p(y|H = 1) &= \mathcal{N}(a, \sigma^2). \end{aligned}$$

*The (log)-likelihood ratio for this problem is*

$$\begin{aligned} L(y) &= \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(y-a)^2}{2\sigma^2} - \log \frac{1}{\sqrt{2\pi\sigma^2}} + \frac{y^2}{2\sigma^2} \\ &= \frac{a^2 - 2ay}{2\sigma^2}. \end{aligned}$$

*Rearranging this, we end up with the following Neyman-Pearson detector for threshold $\nu$:*

$$\delta(y) = \begin{cases} 0 & \text{if } y \leq \frac{\sigma^2}{a}\nu + \frac{a}{2} \\ 1 & \text{if } y > \frac{\sigma^2}{a}\nu + \frac{a}{2} \end{cases}. \tag{3.20}$$

*Observe that we have absorbed the equality case into the LRT, which we can do without loss of performance since this is a continuous detection problem.*

*In this case, the LRT reduces to comparing the received signal $Y$ to a constant threshold value; if $Y$ is greater than the threshold, we suppose $H_1$, and otherwise we suppose $H_0$. In fact, for convenience we will define $y^* := \frac{\sigma^2}{a}\nu + \frac{a}{2}$, and consider the detector to be parameterized by the threshold value rather than the value of the LLR.*

*The Neyman-Pearson lemma tells us that the optimum detector has the form we've just derived, but it doesn't tell us how to choose $\nu$ (or $\eta$ or $y^*$) in order to produce a test of size $\alpha$. Unfortunately, we have to do that for each test individually. In this case, we compute the size of the test as a function of the threshold value $y^*$:*

$$P_F(\delta) = \int_{y^*}^{\infty} p(y|H=0)dy = 1 - \Phi\left(\frac{y^*}{\sigma}\right). \tag{3.21}$$

*Therefore, a test of size $\alpha$ results when we choose $y^*$ as*

$$y^* = \sigma\Phi^{-1}(1-\alpha), \tag{3.22}$$

*where $\Phi^{-1}$ is the* inverse *CDF of the normal distribution. Note that the size does not depend on the $a$, the mean of the alternative hypothesis.*

More complicated hypothesis testing scenarios lead to detectors more complicated than this. We'll look next at multivariate Gaussian signal models. These are ubiquitous in signal processing and machine learning: detecting a frequency-dependent signal in noise, detecting an individual face, and more all can be modeled with reasonable accuracy as choosing between hypotheses with multivariate Gaussian likelihoods. The detectors turn out to be rather different depending on the covariance structure of the Gaussians.

**Example 3.4: Multivariate Gaussian: Different Mean, Same Covariance.** *Let's take the likelihood functions to be multivariate Gaussians:*

$$\begin{aligned} p(\mathbf{y}|H=0) &= \mathcal{N}(\mu_0; \Sigma) \\ p(\mathbf{y}|H=1) &= \mathcal{N}(\mu_1; \Sigma). \end{aligned}$$

*Then, the LLR is*

$$\begin{aligned} L(\mathbf{y}) &= \frac{1}{2}(\mathbf{y}-\mu_0)^T\Sigma^{-1}(\mathbf{y}-\mu_0) - \frac{1}{2}(\mathbf{y}-\mu_1)^T\Sigma^{-1}(\mathbf{y}-\mu_1) \\ &= \mu_1^T\Sigma^{-1}\mathbf{y} - \mu_0^T\Sigma^{-1}\mathbf{y} + \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1) \\ &= (\mu_1-\mu_0)^T\Sigma^{-1}\mathbf{y} + \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1) \end{aligned}$$

*So, the likelihood ratio test gives a detector of the following form*

$$\delta(\mathbf{y}) = \begin{cases} 0 & \text{if } (\mu_1-\mu_0)^T\Sigma^{-1}\mathbf{y} \le \eta \\ 1 & \text{if } (\mu_1-\mu_0)^T\Sigma^{-1}\mathbf{y} > \eta \end{cases}, \tag{3.23}$$

*where we have absorbed the constant term $\frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1)$ into the threshold $\eta$. Testing of Gaussians with the same covariance leads to a* **linear detector**. *We multiply the signal* $\mathbf{y}$ *by a linear operator, and compare the output to a threshold. This simple linear structure is easy to implement and to interpret, and linear classifiers are widely used in machine learning. We'll see linear classifiers again when we study* **logistic regression**.

Things get a bit more complicated when the covariances are different.

**Example 3.5: Multivariate Gaussian: Different Mean, Different Covariance.** *Again take the likelihood functions to be multivariate Gaussians, but with distinct covariance matrices*

$$\begin{aligned} p(\mathbf{y}|H=0) &= \mathcal{N}(\mu_0;\Sigma_0) \\ p(\mathbf{y}|H=1) &= \mathcal{N}(\mu_1;\Sigma_1). \end{aligned}$$

*In this case, the LLR is*

$$\begin{aligned} L(\mathbf{y}) &= \frac{1}{2}(\mathbf{y}-\mu_0)^T\Sigma_0^{-1}(\mathbf{y}-\mu_0) - \frac{1}{2}(\mathbf{y}-\mu_1)^T\Sigma_1^{-1}(\mathbf{y}-\mu_1) \\ &= (\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})\mathbf{y} + \frac{1}{2}(\mathbf{y}^T(\Sigma_0^{-1}-\Sigma_1^{-1})\mathbf{y} + \mu_0^T\Sigma_0^{-1}\mu_0 - \mu_1^T\Sigma_1^{-1}\mu_1). \end{aligned}$$

*This leads to a detector of the form*

$$\delta(\mathbf{y}) = \begin{cases} 0 & \text{if } (\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})\mathbf{y} + \frac{1}{2}(\mathbf{y}^T(\Sigma_0^{-1}-\Sigma_1^{-1}))\mathbf{y} \le \eta \\ 1 & \text{if } (\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})\mathbf{y} + \frac{1}{2}(\mathbf{y}^T(\Sigma_0^{-1}-\Sigma_1^{-1}))\mathbf{y} > \eta \end{cases}, \tag{3.24}$$

*again absorbing the constant term into $\eta$. The first term in the LRT is a linear function of the signal* $\mathbf{y}$*, but the second term is a quadratic term that is inevitable when two signal models have different covariance structure. In fact, in the special case $\mu_0 = \mu_1$, the LRT reduces to computing the quadratic term. We can interpret the quadratic term as testing the signal* $\mathbf{y}$ *against the dominant eigenvectors of $\Sigma_0$; if* $\mathbf{y}$ *lives closer to the dominant eigenvectors of $\Sigma_0$ than to those of $\Sigma_1$, the detector declares $H_0$.*

Finally, we'll examine the detector associated with the email spam filter.

**Example 3.6: Spam Detection.** *As described above, in this case the received "signal" is the $n$-dimensional vector of word counts of an $n$-word email message, and the resulting likelihood functions follow a multinomial distribution:*

$$p(\mathbf{y}|H=0) = \frac{n!}{y_1!\cdots y_k!}(p_1^0)^{y_1}\cdot(p_2^0)^{y_2}\cdots(p_k^0)^{y_k} \tag{3.25}$$

$$p(\mathbf{y}|H=1) = \frac{n!}{y_1!\cdots y_k!}(p_1^1)^{y_1}\cdot(p_2^1)^{y_2}\cdots(p_k^1)^{y_k}. \tag{3.26}$$

*Despite the complicated form of the likelihood functions, the log-likelihood ratio is quite simple.*

$$\begin{aligned} L(\mathbf{y}) &= \sum_{i=1}^k y_i \log p_i^1 - \sum_{i=1}^k y_i \log p_i^0 \\ &= \sum_{i=1}^k y_i \log \frac{p_i^1}{p_i^0}. \end{aligned}$$

*The LRT leads to the following detector:*

$$\delta(\mathbf{y}) = \begin{cases} 0 & \text{if } \sum_{i=1}^k y_i \log \frac{p_i^1}{p_i^0} \le \eta \\ 1 & \text{if } \sum_{i=1}^k y_i \log \frac{p_i^1}{p_i^0} > \eta \end{cases}. \tag{3.27}$$

*Although it may not be obvious at first glance, this too is a linear detector! We multiply the vector of word counts by the vector of log probabilities. Roughly speaking, this detector tells us whether the word counts are closer to what's expected from a regular message or a spam message.*

### 3.1.2 Receiver Operator Characteristic

The Neyman-Pearson framework cannot tell us which is the "correct" size for a hypothesis test. In experimental science, standards of $\alpha = 0.05$ or $\alpha = 0.01$ are common thresholds of significance. These standards are often the subject of controversy, and certainly they are not universally applicable.

The **receiver operator characteristic** (ROC) curve is a universal way of characterizing the performance in a hypothesis testing scenario. The ROC takes its name from radar detection, where system designers wanted to characterize the performance of a radar receiver. Rather than pick a particular operating point, the ROC shows the trade-off between the false alarm probability $P_F(\delta)$ and the detection probability $P_D(\delta)$ of the Neyman-Pearson detector $\delta$ associated with every $P_F(\delta) = \alpha$. We usually plot $\alpha$ on the x-axis and $P_D(\delta)$ of the resulting detector on the y-axis.

In most cases, the ROC curve has two endpoints: $(0, 0)$ and $(1, 1)$. If we want zero false alarms, we must also settle for zero correct detections. If we allow for all the false alarms, we will never miss a detection. One can construct pathological examples in which these are not the endpoints, but they are not of much practical interest.

Every ROC curve is convex. To see this, consider two sizes $\alpha_1, \alpha_2$, and let $P_{D1}, P_{D2}$ be the detection probabilities of the optimum Neyman-Pearson detector. We can always use the detector that chooses randomly between the two Neyman-Pearson detectors. By doing so we can achieve any trade-off on the line segment that connects $(\alpha_1, P_{D1})$ and $(\alpha_2, P_{D2})$. The Neyman-Peason detector can only do better than this randomized rule, so the ROC curve cannot be lower than this line segment.

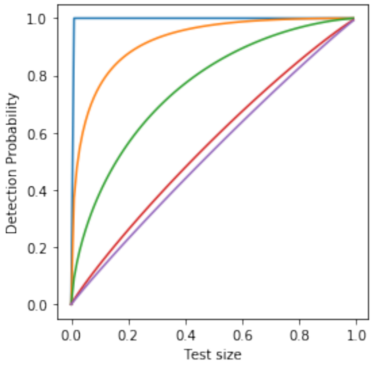

Figure 3.1: The receiver operator characteristic for the simple detector with $a = 1$ and $\sigma^2 \in \{0.1, 0.5, 1, 5, 10\}$.

**Example 3.7: Simple Object Detection.** *We saw above that for the detector with threshold $y^*$, the size of the test is*

$$P_F(\delta) = \int_{y^*}^{\infty} p(y|H = 0)dy = 1 - \Phi\left(\frac{y^*}{\sigma}\right). \tag{3.28}$$

*We can compute the ROC curve of this scenario by deriving the detection probability:*

$$P_D(\delta) = \int_{y^*}^{\infty} p(y|H = 1)dy = 1 - \Phi\left(\frac{y^* - a}{\sigma}\right). \tag{3.29}$$

*Substituting the $y^*$ associated with size $\alpha$, we get the trade-off*

$$P_D(\alpha) = 1 - \Phi\left(\Phi^{-1}(1 - \alpha) - \frac{a}{\sigma}\right). \tag{3.30}$$

*Again we cannot evaluate this in closed form, but the resulting function is decreasing in $\alpha$ and $\sigma$, but increasing in $a$. As the variance increases, we get a worse trade-off between power and size, but as the mean increases, we get a better one. In Figure 3.1 we plot the ROC curve for $a = 1$ and for $\sigma^2 \in \{0.1, 0.5, 1, 5, 10\}$.*

### 3.1.3 Bayesian Hypothesis Testing

In Bayesian hypothesis testing, we get to sidestep the question of false alarm vs. missed detection probability entirely. We do this by making a crucial assumption: that $H$ is a random variable whose distribution $p(h)$ is known to us. We call this the **prior distribution** on $H$, because it is the distribution over the hypotheses *before* we have seen any evidence in the form of the signal $Y$. In the binary case, we express the prior by writing

$$\begin{aligned}\pi_0 &:= p(H=0)\\ \pi_1 &:= p(H=1),\end{aligned}$$

where of course $\pi_0 = (1-\pi_1)$ so that $p(h)$ is a valid pmf.

The availability of the prior is controversial, which is why many engineers, statisticians, and scientists prefer to eschew Bayesian probability altogether. But knowledge of the prior is often perfectly justified. In radar detection, we may know the density or frequency of objects, from which we can derive a probability of seeing one even before we take an observation $Y$. In spam filtering, it is not hard to estimate the ratio of spam to real messages. In detecting objects on the road, we can study how often a pedestrian, another car, or a traffic signal will show up on the average. In medical experiments, we may know from scientific first principles—the result of previous biochemical analysis, for example—whether a drug is likely to have an impact. In these cases, there seems to be no problem with supposing access to the prior. Alternatively, even in cases where we don't know the prior, we may regard it as methodological assumption, or even a hypothetical: *Supposing* that the prior probabilities are $\pi_0$ and $\pi_1$, what is the optimum detector?

Let's answer this question in the Bayesian framework. As before, the detector $\delta(y)$ is defined entirely by a partition of $\Gamma$ into sets $\Gamma_0$ and $\Gamma_1$. To choose the optimum detector, we first need to define our design criteria! In the Bayesian case, we are fortunate enough to have a single metric by which to judge success: the **probability of detection error**. Define a random variable associated with an error:

$$E = \begin{cases} 0 & \text{if } \delta(Y) = H \\ 1 & \text{if } \delta(Y) \neq H \end{cases}. \tag{3.31}$$

In other words, $E = 0$ if the detector correctly determines the state of nature from $Y$, and is equal to 1, indicating an error, otherwise That is, we have an error if we get either a Type I or Type II error. We can compute the probability of error by noting that each error event corresponds with the events $Y \in \Gamma_0$ or $Y \in \Gamma_1$:

$$p(E=1|H=1) = \int_{\Gamma_0} p(y|H=1)dy \tag{3.32}$$

$$p(E=1|H=0) = \int_{\Gamma_1} p(y|H=0)dy \tag{3.33}$$

. Then, noting that the joint probability mass function factors $p(e,h) = p(e|h)p(h)$, and marginalizing out $h$, we get: using the law of total probability:

$$p(E=1) = p(E=1|H=0)\pi_0 + p(E=1|H=1)\pi_1 \tag{3.34}$$

$$= \pi_0 \int_{\Gamma_1} p(y|H=0)dy + \pi_1 \int_{\Gamma_0} p(y|H=1)dy. \tag{3.35}$$

The region of integration for each of the terms in (3.35) comes from the definition of a detection error. Supposing that $H = 1$, we compute the probability that $\delta(Y) = 0$ by integrating $p(y|H=1)$ over the region for which $\delta(Y) = 0$, and vice versa for $H = 0$.

The expression in (3.35) may look discouraging in view of our objective. We have to optimize over the *regions of integration* to minimize the probability of detection error. Our usual optimization procedures offer little help: how do we take derivatives with respect to sets?

Fortunately, a simple argument provides an intuitive solution to the puzzle. Here's the key fact: *every point in* $\Gamma$ *must be assigned either to* $\Gamma_0$ *or* $\Gamma_1$. Instead of trying to optimize over the sets $\Gamma_0$ and $\Gamma_1$ directly, let's examine points in $y \in \Gamma$ individually and ask: is the probability of error lower if this point belongs to the first integral in (3.35) or the second? We can answer this by inspecting the integrands: if $\pi_0 p(y|H = 0) > \pi_1 p(y|H = 1)$, then $y$ contributes less to the probability of error if it is a member of $\Gamma_0$; otherwise it should be a member of $\Gamma_1$. When $\pi_0 p(y|H = 0) = \pi_1 p(y|H = 1)$, we can break the tie arbitrarily, and our decision will not impact the detection error probability.

We can write down the optimum detector like this:

$$\delta(y) = \begin{cases} 0, & \text{if } \frac{p(y|H=1)}{p(y|H=0)} < \frac{\pi_0}{\pi_1} \\ 1, & \text{if } \frac{p(y|H=1)}{p(y|H=0} \geq \frac{\pi_0}{\pi_1} \end{cases}. \tag{3.36}$$

Equivalently, we can define the detector $\delta$ in terms of the sets $\Gamma_0$ and $\Gamma_1$:

$$\Gamma_0 = \left\{ y : \frac{p(y|H = 1)}{p(y|H = 0)} < \frac{\pi_0}{\pi_1} \right\} \tag{3.37}$$

$$\Gamma_1 = \left\{ y : \frac{p(y|H = 1)}{p(y|H = 0} \geq \frac{\pi_0}{\pi_1} \right\}. \tag{3.38}$$

Observe that, just as in the Neyman-Pearson framework, the optimum detector has the form of an LRT. Bayes-optimum detection entails comparing the likelihood ratio $l(y)$ to a threshold determined by the prior probabilities. If the priors are uniform ($\pi_0 = \pi_1 = 0.5$), the test reduces to choosing the hypothesis with maximum likelihood. Otherwise, the priors bias the test: The smaller $\pi_0$ is, the larger the likelihood $p(y|H = 0)$ must be for the optimum detector to return $\delta(y) = 0$.

Similar to the Neyman-Pearson test, we can write the Bayes detector in terms of the log-likelihood ratio $L(y)$:

$$\delta(y) = \begin{cases} 0, & \text{if } L(y) < \log \frac{\pi_0}{\pi_1} \\ 1, & \text{if } L(y) < \log \frac{\pi_0}{\pi_1} \end{cases}. \tag{3.39}$$

**Example 3.8: Simple Object Detection.** *Let's (re-)revisit the simple detection system in which* $H_0$ *and* $H_1$ *have the following likelihood functions:*

$$p(y|H = 0) = \mathcal{N}(0, \sigma^2)$$
$$p(y|H = 1) = \mathcal{N}(a, \sigma^2).$$

*Recall that the LLR is*

$$L(y) = \frac{a^2 - 2ay}{2\sigma^2},$$

*so the Bayes-optimum detector is*

$$\delta(y) = \begin{cases} 0 & \textit{if } y \leq \frac{\sigma^2}{a} \log \frac{\pi_0}{\pi_1} + \frac{a}{2} \\ 1 & \textit{if } y > \frac{\sigma^2}{a} \log \frac{\pi_0}{\pi_1} + \frac{a}{2} \end{cases}. \tag{3.40}$$

*Observe that when the priors are equal ($\pi_0 = \pi_1 = 1/2$), the threshold value becomes $y^* = 1/2$. This makes perfect sense: if the hypotheses are equally likely* a priori, *the optimum choice is the mean value closest to the received signal $Y$.*

Generally speaking, all of the Neyman-Pearson examples we considered above can be re-cast in the Bayesian framework, with the likelihood ratio (resp. LLR) thresholds $\eta$ (resp. $\nu$) replaced by the ratio of the prior probabilities. As $\pi_0 \to 0$, the threshold goes to zero (or to $-\infty$ for the LLR), and the detector always chooses $\delta(y) = 1$—as it should; this prior indicates *certainty* that the null hypothesis is true! Similarly, as $\pi_0 \to 1$, the detector always chooses $\delta(y) = 1$.

We will not revisit all of the Neyman-Pearson examples considered above, but one example is illustrative.

**Example 3.9: Multivariate Gaussian: Different Mean, Isotropic Covariance.** *Let's take the likelihood functions to be multivariate Gaussians:*

$$\begin{aligned} p(\mathbf{y}|H=0) &= \mathcal{N}(\mu_0, \sigma^2\mathbf{I}) \\ p(\mathbf{y}|H=1) &= \mathcal{N}(\mu_1, \sigma^2\mathbf{I}). \end{aligned}$$

*In this case we say that the covariance is* **isotropic***, meaning that the variance is equal in all directions. Equivalently, the noise variance is equal for every element, and the noise elements are independent. The LLR for this special case is:*

$$\begin{aligned} L(\mathbf{y}) &= \frac{1}{2\sigma^2}(\mathbf{y}-\mu_0)^T(\mathbf{y}-\mu_0) - \frac{1}{2\sigma^2}(\mathbf{y}-\mu_1)^T(\mathbf{y}-\mu_1) \\ &= \frac{1}{\sigma^2}(\mu_1^T\mathbf{y} - \mu_0^T\mathbf{y}) + \frac{1}{2\sigma^2}(\mu_0^T\mu_0 - \mu_1^T\mu_1) \\ &= \frac{1}{\sigma^2}(\mu_1-\mu_0)^T\mathbf{y} + \frac{1}{2\sigma^2}(\|\mu_0\|^2 - \|\mu_1\|^2). \end{aligned}$$

*Let's further suppose equal priors, i.e.* $\pi_0 = \pi_1 = 1/2$*. Then, the threshold for detection is* $L(\mathbf{y}) = 0$*, and the Bayes-optimum detector is*

$$\delta(\mathbf{y}) = \begin{cases} 0 & \text{if } (\mu_1-\mu_0)^T\mathbf{y} \le \frac{1}{2}(\|\mu_1\|^2 - \|\mu_0\|^2) \\ 1 & \text{if } (\mu_1-\mu_0)^T\mathbf{y} < \frac{1}{2}(\|\mu_1\|^2 - \|\mu_0\|^2) \end{cases}, \tag{3.41}$$

*In this case, the linear classifier that emerges from the problem draws a hyperplane equidistant between* $\mu_0$ *and* $\mu_1$*, and orthogonal to the line connecting them. The division of two classes via a hyperplace is commonplace in machine learning. In this special case, we can also see that the classifier is equivalent simply to checking whether* $\|\mathbf{y}-\mu_0\|$ *is smaller than* $\|\mathbf{y}-\mu_1\|$ *and declaring* $\delta(\mathbf{y}) = 0$ *if so.*

#### 3.1.3.1 Minimizing Bayes Risk

We may be tempted to conclude that this formulation is the end of the story as far as hypothesis testing goes. However, earlier we claimed that no single approach will settle conclusively the trade-offs inherent to hypothesis testing. Here, we have smuggled in a perhaps subtle assumption: all detection errors are equally "bad." Minimizing the probability of detection makes sense when mistaking $H_0$ for $H_1$ is just as costly as mistaking $H_1$ for $H_0$. Indeed, it may be much worse to label incorrectly an email as spam, because we may miss reading an important message, than it is to miss the detection of an email message.

If we can quantify the costs for different types of mistakes, we can incorporate them into the Bayesian framework via **Bayes risk minimization**.

**Definition 3.3:.** *For a binary hypothesis testing problem, the* **cost function** *is denoted by four values $C_{00}$, $C_{10}$, $C_{01}$, and $C_{11}$, which are arbitrary except for the constraints $C_{01} > C_{11}$ and $C_{10} > C_{10}$. These quantities indicate the relative cost of each of the four possible outcomes: $H_0$ is true, and $\delta(Y) = 0$, $H_0$ is true, but $\delta(Y) = 1$, and so forth. For a detector $\delta$, the* **Bayes Risk** *is defined as*

$$\begin{aligned} r(\delta) = E[C] = C_{00}\text{Pr}(H_0, \delta(Y) = 0) + C_{10}\text{Pr}(H_0, \delta(Y) = 1) + \\ C_{01}\text{Pr}(H_1, \delta(Y) = 0) + C_{11}\text{Pr}(H_1, \delta(Y) = 1). \end{aligned} \tag{3.42}$$

This risk function is in fact an expectation of the random variable $C(H, E)$, which has the four values assigned under the four possible circumstances.

If $Y$ is continuous, we can expand $r(\delta)$ in terms of integrals over the likelihood functions. A similar expression, with sums instead of integrals, holds for discrete $Y$.

$$r(\delta) = C_{00}\text{Pr}(H_0, \delta(Y) = 0) + C_{10}\text{Pr}(H_0, \delta(Y) = 1) + C_{01}\text{Pr}(H_1, \delta(Y) = 0) + C_{11}\text{Pr}(H_1, \delta(Y) = 1) \tag{3.43}$$

$$= C_{00}\pi_0 \int_{\Gamma_0} p(y|H_0)dy + C_{10}\pi_0 \int_{\Gamma_1} p(y|H_0)dy + C_{01}\pi_1 \int_{\Gamma_0} p(y|H_1)dy + C_{11}\pi_1 \int_{\Gamma_1} p(y|H_1)dy \tag{3.44}$$

$$= \int_{\Gamma_0} (C_{00}\pi_0 p(y|H_0) + C_{01}\pi_1 p(y|H_1))dy + \int_{\Gamma_1} (C_{10}\pi_0 p(y|H_0)dy + C_{11}\pi_1 p(y|H_1))dy. \tag{3.45}$$

Similar to the minimization of detection error probability, we can minimize the Bayes risk by looking at each point $y \in \Gamma$ individually and choosing the region $\Gamma_0$ or $\Gamma_1$ for which the corresponding integrand is smallest. By this reasoning the optimum choice of $\Gamma_0$ is

$$\Gamma_0 = \{y \in \mathcal{G} : C_{00}\pi_0 p(y|H_0) + C_{01}\pi_1 p(y|H_1) < C_{10}\pi_0 p(y|H_0)dy + C_{11}\pi_1 p(y|H_1)\} \tag{3.46}$$

$$= \{y \in \mathcal{G} : (C_{00} - C_{10})\pi_0 p(y|H_0) < (C_{11} - C_{01})\pi_1 p(y|H_1)\} \tag{3.47}$$

$$= \left\{ y : \frac{p(y|H_1)}{p(y|H_0)} < \frac{(C_{10} - C_{00})\pi_0}{(C_{01} - C_{11})\pi_1} \right\}, \tag{3.48}$$

and of course $\Gamma_1 = \Gamma \setminus \Gamma_0$. Then, the optimum detector can be written as

$$\delta(y) = \begin{cases} 0, & \text{if } l(y) < \frac{(C_{10}-C_{00})\pi_0}{(C_{01}-C_{11})\pi_1} \\ 1, & \text{if } l(y) \geq \frac{(C_{10}-C_{00})\pi_0}{(C_{01}-C_{11})\pi_1} \end{cases}. \tag{3.49}$$

Let's pause to examine a few facts. First, Bayes risk minimization boils down to a LRT, just as did the minimization of detection error probability. Here the threshold for the LRT depends on both the priors and the cost assignments. Second, if we choose $C_{00} = C_{11} = 0$ and $C_{10} = C_{01} = 1$, the LRT threshold reduces to that of minimization of detection error probability. This corresponds to our earlier intuition that minimizing the probability of detection error is appropriate when both types of error are equally "bad." Finally, note that the LRT threshold is invariant to scales and shifts of the cost assignments. The threshold depends on the ratio of the difference, so if we add a constant to each cost or multiply each cost by a constant, the threshold is unchanged. Thus the cost assignments are expressions of *relative* cost, we don't need to express them in absolute units.

This means that we can choose $C_{00} = C_{11} = 0$ without loss of generality. Then, the threshold depends on the ratio of the priors and the ratio of the error costs $C_{01}$ and $C_{10}$.

Because this framework again boils down to an LRT, we won't belabor the point and go through all of the previous examples again. But we'll check in with the scalar object detection to build up our intuition.

**Example 3.10: Object Detection Revisited.** *Let's take yet another look at the object detection example above under the Bayes risk framework. Let $C_{01}$ and $C_{10}$ be the costs associated with missing a detection and falsely declaring an alarm, respectively, and for simplicity let $C_{00} = C_{11} = 0$. What detector minimizes the Bayes risk?*

*Fortunately, we can see by inspection that the new LRT threshold is differs from the old threshold only in that we multiply the ratio of the prior by the ratio of the cost assignments. We get the detector defined by the following region for $H_0$:*

$$\Gamma_0 = \left\{ y : y < \frac{a}{2} + \frac{\sigma^2}{a} \log\left(\frac{C_{10}\pi_0}{C_{01}\pi_1}\right) \right\}. \tag{3.50}$$

*Similar to before, the "default" threshold—which obtains if the priors are uniform and the costs are equal—is $a/2$, and the LRT reduces to deciding whether $Y$ is closer to $0$ or $a$. For non-uniform priors and unequal cost assignments, we bias the detector in the direction of the hypothesis that is more probable* a priori *and/or more costly to guess wrong. If $C_{10}$ is substantially larger than $C_{01}$, we take care to declare $H_1$ as the hypothesis only when we are* very *sure that we are correct!*

The Bayesian framework is powerful, and it provides clear answers so long as we have all of the inputs. If we know the "true" priors $\pi_0$ and $\pi_1$, and if there are unambiguous cost assignments—not to mention the likelihood functions $p(y|H = 0)$ and $p(yH = 1)$!—we can solve for the detector that minimizes the Bayes risk and which is optimum with respect to this framework. But in practice we will often not have all of this information. Priors need to be estimated from previous encounters with the same phenomenon—what if we are testing a hypothesis with which no one has much experience? Similarly, there rarely are clear-cut cost assignments. Can you quantify how much worse (or better) it is to guess incorrectly $H_0$ than $H_1$? Instead of providing a single "best" detector, the Bayesian framework provides a way to think systematically about choosing one. It cannot provide priors or cost functions, but it can tell us the form of the detector and how it depends on the prior we assume and the cost assignments we choose.

In the next section we will examine an extremely conservative approach to detection, which does not require us to know priors or even to decide on an acceptable false-alarm probability.

### 3.1.4 Minimax Hypothesis Testing

When priors $\pi_0$ and $\pi_1$ are unavailable, we need a different framework for choosing a detector. Here we'll examine the **minimax** criterion, which is a conservative, risk-averse approach. Keep the same framework as before: likelihood functions, detector definitions, cost assignments, etc., but suppose there is no (known) prior probability over the hypotheses $H_0$ and $H_1$. Even without priors, we still can talk about the expected costs *given* that $H_0$ or $H_1$ is true.

**Definition 3.4:.** *For a detector $\delta$ defined by the decision regions $\Gamma_0$ and $\Gamma_1$, define the* **conditional**

**risks** $R_0(\delta)$ *as the expected cost given* $H_0$ *and* $R_1(\delta)$ *as the expected cost given* $H_1$*, as follows:*

$$R_0(\delta) = C_{00}\Pr(\delta(Y) = 0|H = 0) + C_{10}\Pr(\delta(Y) = 1|H = 0) \tag{3.51}$$
$$R_1(\delta) = C_{01}\Pr(\delta(Y) = 0|H = 1) + C_{11}\Pr(\delta(Y) = 1|H = 1). \tag{3.52}$$

It's easy to see that if there exists a known prior, we can recover the Bayes risk by averaging over $R_0(\delta)$ and $R_1(\delta)$:

$$r(\delta) = \pi_0 R_0(\delta) + \pi_1 R_1(\delta). \tag{3.53}$$

If we don't know or don't want to assume a prior, we can take a *minimax* or *worst-case* approach to risk minimization, in which we choose the detector that *minimizes* the *maximum* conditional risk. (Hence the name!) Formally, we define the minimax cost function as follows.

**Definition 3.5:.** *For a binary hypothesis test and a detector* $\delta$ *with conditional risks* $R_0(\delta)$ *and* $R_1(\delta)$*, the* **worst-case conditional risk** *is*

$$s(\delta) = \max\{R_0(\delta), R_1(\delta)\}. \tag{3.54}$$

Then, the objective is to find the detector $\delta(y)$ that minimizes $s(\delta)$.

One way to think of minimax detection is in terms of Bayesian detection under the worst-case *prior*. You will see this referred to in the literature as *adversarial detection.* Imagine that you choose a detector $\delta$, after which an "adversary" gets to choose a prior $\pi_0, \pi_1$ to make the Bayes risk as large as possible. In this scenario, the best strategy is to choose the detector that minimizes the risk of the worst-case prior.

It turns out that this scenario is equivalent to minimax detection. Let's find out why.

**Definition 3.6:.** *For a binary hypothesis detection problem and detector* $\delta$*, the* **worst-case Bayes risk** *is*

$$t(\delta) = \max_{\pi_0} \pi_0 R_0(\delta) + (1 - \pi_0) R_1(\delta). \tag{3.55}$$

Then, we find the detector that minimizes $t(\delta)$.

First, define the following function:

$$V(\pi_0) = \min_{\delta} r(\delta; \pi_0) = \min_{\delta} \pi_0 R_0(\delta) + (1 - \pi_0) R_1(\delta). \tag{3.56}$$

This function expresses the smallest Bayes risk for a particular "choice" of prior $\pi_0$. An important fact about $V(\pi_0)$ is that it is concave, which we show in the following lemma.

**Lemma 3.2:.** *The function* $V(\pi_0)$ *is concave, i.e.*

$$V(\alpha\pi_0 + (1 - \alpha)\pi_0') \geq \alpha V(\pi_0) + (1 - \alpha) V(\pi_0'), \tag{3.57}$$

*for any* $0 \leq \alpha, \pi_0, \pi_0' \leq 1$.

*Proof.* We see this by bounding the function for a convex combination of priors $\pi_0$ and $\pi_0'$. For

$0 \leq \alpha \leq 1$, we have

$$V(\alpha\pi_0 + (1-\alpha)\pi_0') = \min_\delta \alpha\pi_0 R_0(\delta) + \alpha(1-\pi_0)R_1(\delta) + (1-\alpha)\pi_0' R_0(\delta) + (1-\alpha)(1-\pi_0')R_1(\delta) \tag{3.58}$$

$$\geq \alpha\left[\min_\delta \pi_0 R_0(\delta) + (1-\pi_0)R_1(\delta)\right] + (1-\alpha)\left[\min_\delta \pi_0' R_0(\delta) + (1-\pi_0')R_1(\delta)\right] \tag{3.59}$$

$$= \alpha V(\pi_0) + (1-\alpha)V(\pi_0'). \tag{3.60}$$

□

The concavity of $V(\pi_0)$ implies the fact we're trying to establish: the minimax detector is the Bayes-optimum detector for some $\pi_0$. We lay the argument out in the following theorem.

**Theorem 3.1:.** *For a binary hypothesis test, the minimax detector $\delta(y)$ also minimizes the Bayes risk for some prior $\pi_0$.*

*Proof.* To see this, consider a the Bayes risk of a fixed detector $\delta(y)$ as a function of the prior $\pi_0$. For $\pi_0 = 0$, the Bayes risk is $R_1(\delta)$, for $\pi_0 = 1$, the Bayes risk is $R_0(\delta)$, and for any $0 < \pi_0 < 1$ the Bayes risk is the convex combination of $R_0(\delta)$ and $R_1(\delta)$. In other words, we can plot the Bayes risk of $\delta(y)$ by plotting the line that has $R_0(\delta)$ and $R_1(\delta)$ as endpoints.

To find the minimax detector $\delta(y)$, we want to make $R_0(\delta)$ and $R_1(\delta)$ as small as possible. However, we cannot reduce $R_0(\delta)$ and $R_1(\delta)$ so far that the line joining them lies beneath $V(\pi_0)$. Therefore, the minimax detector has a Bayes risk curve that intersects $V(\pi_0)$. Since $V(\pi_0)$ is concave, that Bayes risk curve intersects $V(\pi_0)$ at exactly one point, meaning that the minimax detector is the Bayes-optimum detector for that point. □

But *which* Bayes-optimum detector? Intuitively, we expect the minimax detector to correspond to the worst-case Bayes risk, i.e. the detector associated with the $\pi$ that maximizes $V(\pi_0)$. When $V(\pi_0)$ is differentiable, this intuition is correct. As far as minimax detection goes, there are three possibilities: (1) the minimax detector has $R_0(\delta) < R_1(\delta)$, (2) the minimax detector has $R_1(\delta) < R_0(\delta)$, and (3) the minimax detector has $R_1(\delta) = R_0(\delta)$.

When $V(\pi_0)$ is differentiable and non-constant, each one of these cases corresponds to a different location of the maximizing value $\pi^* = \arg\max_\pi V(\pi)$. Say $\pi_0^* = 0$. Clearly the optimum detector is the one that minimizes the Bayes risk for $\pi_0 = 1$, and $R_0(\delta) < R_1(\delta)$. The obverse is true if $\pi_0 = 1$. For $0 < \pi^* < 1$, we have $V'(\pi^*) = 0$. The Bayes risk curve follows the tangent line at $\pi^*$, which has zero slope, and thus $R_0(\delta) = R_1(\delta)$.

**Example 3.11: Simple Object Detection.** *Let's again consider the simple object/anomaly detection problem in which $H_0$ and $H_1$ have the following likelihood functions:*

$$p(y|H=0) = \mathcal{N}(0, \sigma^2) \tag{3.61}$$

$$p(y|H=1) = \mathcal{N}(a, \sigma^2), \tag{3.62}$$

*and we have error costs $C_{01}$ and $C_{10}$. To find the minimax detector, we'll consider the Bayes risk of the optimum detector for each prior $\pi_0$. Recall from the previous section that the LRT resulted* *in a detector that thresholds the received signal $y$.*

$$\delta(y) = \begin{cases} 0, & y \leq \frac{\sigma^2}{a} \log\left(\frac{C_{10}\pi_0}{C_{01}\pi_1}\right) \\ 1, & y > \frac{a}{2} + \frac{\sigma^2}{a} \log\left(\frac{C_{10}\pi_0}{C_{01}\pi_1}\right). \end{cases} \tag{3.63}$$

*Let's compute $V(\pi_0)$ for this scenario:*

$$\begin{aligned} V(\pi_0) &= \pi_0 C_{10} \int_{y^*(\pi_0)}^{\infty} p(y|H=0)dy + (1-\pi_0)C_{01} \int_{-\infty}^{y^*(\pi_0)} p(y|H=1)dy && (3.64) \\ &= \pi_0 C_{10}(1 - \Phi\left(\frac{y^*(\pi_0)}{\sigma}\right) + (1-\pi_0)C_{01}\Phi\left(\frac{y^*(\pi_0)-a}{\sigma}\right). && (3.65) \end{aligned}$$

*where $y^*(\pi_0)$ is the detector threshold for the prior $\pi_0$. To find the optimum value of $\pi_0$, we need to differentiate $V(\pi_0)$ and set it equal to zero. This does not have a closed-form solution; instead we solve numerically for the optimum value. In the special case of uniform costs, $C_{10} = C_{01} = 1$, we get*

$$\pi_0(1 - \Phi\left(\frac{y^*(\pi_0)}{\sigma}\right) + (1-\pi_0)\Phi\left(\frac{y^*(\pi_0)-a}{\sigma}\right). \tag{3.66}$$

*Because $V(\pi_0)$ is differentiable, the solution lies on the interior of $[0, 1]$, which means that the solution corresponds to $R_0 = R_1$. Thus, the solution satisfies*

$$(1 - \Phi\left(\frac{y^*(\pi_0)}{\sigma}\right) = \Phi\left(\frac{y^*(\pi_0)-a}{\sigma}\right). \tag{3.67}$$

*By inspection, the solution is $y^* = a/2$.*

This result should strike you as intuitively pleasing: the prior that results in the minimax detector, which we interpret as the worst-case prior, is the **uniform prior**, the one that tells us as little as possible *a priori* which hypothesis is true. In Bayesian statistics, there is an entire body of research on choosing **non-informative priors**, or prior distributions $p(h)$ that make minimal assumptions about $H$. The worst-case prior is a special case of a non-informative prior, and in many cases the prior indeed turns out to be uniform.

## 3.2 Multiple Hypothesis Testing

So far, we have considered only *binary* hypothesis testing. In this section we consider $M > 2$ hypotheses, and try to generalize the techniques developed in previous sections. As we will see, there is often no straightforward way of doing so.

Similar to before, we observe a random variable $Y$, which may be continuous or discrete. Also similar to before, the distribution of $Y$ depends on the state of nature $H$, which corresponds to which hypothesis is true. However, instead of being a binary random variable, $H \in \{0, 1, \ldots M-1\}$ takes one of $M$ values. Indeed, each value of $H$ corresponds to a hypothesis, and we have a likelihood function associated with each value of $H$:

$$\begin{aligned} H_0 &: Y \sim p(y|H=0) && (3.68)\\ H_1 &: Y \sim p(y|H=1) && (3.69)\\ &\vdots && (3.70)\\ H_{M-1} &: Y \sim p(y|H=M-1). && (3.71) \end{aligned}$$

In this case it does not make sense to consider $H_0$ to be the null hypothesis or for the remaining hypotheses to be "alternative" hypotheses. There are simply $M$ possible states of nature to choose from, corresponding to $M$ different distributions on the observation $Y$.

### 3.2.1 Bayesian Multiple Hypothesis Testing

If we suppose the Bayes framework, we *can* derive an optimum detector. Similar to before, define a prior distribution over the $M$ hypotheses

$$\begin{aligned} \pi_0 &= p(H=0) && (3.72)\\ \pi_1 &= p(H=1) && (3.73)\\ &\vdots && (3.74)\\ \pi_{M-1} &= p(H=M-1). && (3.75) \end{aligned}$$

Again we define a detector $\delta(y)$, which this time maps from the support of $Y$ to the set $\{0, 1, \ldots, M-1\}$. We express the detector in terms of sets $\Gamma_0, \Gamma_1, \ldots, \Gamma_{M-1}$ that form a partition of the support of $Y$. Then, the detector is simply

$$\delta(y) = \begin{cases} 0, & y \in \Gamma_0 \\ 1, & y \in \Gamma_1 \\ \vdots & \\ M-1, & y \in \Gamma_{M-1} \end{cases}. \tag{3.76}$$

We also define a cost function. Here we have to take extra care, because there are $M^2$ possible costs to consider. If $H_0$ is true but we guess $H_1$, we may suffer a different cost than if we guess $H_2$. Therefore, define the cost assignment

$$C_{ij} = \text{The cost suffered if } \delta(Y) = i \text{ and } H_j \text{ is true.} \tag{3.77}$$

Then, we can define the Bayes risk:

$$\begin{aligned} r(\delta) &= \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} C_{ij} \Pr(\delta(Y) = y, H = j) && (3.78)\\ &= \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \pi_j \int_{\Gamma_i} p(y|H=j) dy, && (3.79) \end{aligned}$$

supposing that $Y$ is continuous. If $Y$ is discrete, then the integrals in the previous expression become sums over the sets $\Gamma_i$.

We'd like to find the detector that minimizes the Bayes risk. Unfortunately, we cannot compute a likelihood ratio test—we have more than two likelihoods! Instead, we need to consider the argument that we made when we first derived the LRT for Bayesian hypothesis testing. There, we asked the question: for each value of $y$, what set $\Gamma_i$ has the *smallest integrand*? If we put $y$ in that set, we minimize the Bayes risk. This is a bit easier to see if we rewrite the Bayes risk somewhat:

$$r(\delta) = \sum_{i=0}^{M-1} \int_{\Gamma_i} \sum_{j=1}^{M-1} C_{ij}\pi_j p(y|H=j)dy. \tag{3.80}$$

From this expression, it is "easy" to pick out the optimum detector. We put $y \in \Gamma_i$ if the integrand associated with $H_i$ is smaller than that of any other $H_k$. In equations, this means that

$$\Gamma_i = \left\{ \sum_{j=1}^{M-1} C_{ij}\pi_j p(y|H=j) \leq C_{kj} \sum_{j=1}^{M-1} \pi_j p(y|H=j), \forall k \neq i \right\}. \tag{3.81}$$

Equivalently, we can write that

$$\delta(y) = \arg\min_i \sum_{j=1}^{M-1} C_{ij}\pi_j p(y|H=j). \tag{3.82}$$

It's easy to convince yourself that this collapses to the Bayes-optimum LRT in the case of $M = 2$. Unfortunately, it's difficult to get much intuition beyond this, and it's rather challenging to compute or to visualize the decision regions.

Instead, let's consider the simple case in which $C_{ij} = 1$ if $i \neq j$, and $C_{ij} = 0$ for $i = j$. That is, we suffer no cost if the detector returns the correct hypothesis, and uniform costs otherwise. In this case, the Bayes risk is equal to the probability of error:

$$r(\delta) = \sum_{i=0}^{M-1} \int_{\Gamma_i} \sum_{j\neq i} \pi_j p(y|H=j)dy. \tag{3.83}$$

Here, the optimum detector is easier to pick out. We can restate the optimum detector using (3.82):

$$\delta(y) = \arg\min_i \sum_{j\neq i} \pi_j p(y|H=j). \tag{3.84}$$

To get a little more intuition, notice that for every $i$, all but one of the terms in the sum is the same. Also, for a fixed $y$, all $M$ terms $\pi_j p(y|H=j)$ sum to a constant:

$$\sum_j \pi_j p(y|H=j) = p(y). \tag{3.85}$$

Instead of looking at which $i$ has the smallest sum $\sum_{j\neq i} \pi_j p(y|H=j)$, we can look instead at which $i$ has the *largest* term $\pi_i p(y|H = i)$:

$$
\begin{aligned}
\delta(y) &= \arg\min_i \sum_{j \neq i} \pi_j p(y|H = j) && (3.86)\\
&= \arg\min_i \sum_{j=0}^{M-1} \pi_j p(y|H = j) - \pi_i p(y|H = i) && (3.87)\\
&= \arg\min_i p(y) - \pi_i p(y|H = i) && (3.88)\\
&= \arg\min_i -\pi_i p(y|H = i). && (3.89)\\
&= \arg\max_i \pi_i p(y|H = i), && (3.90)
\end{aligned}
$$

where we can drop the $p(y)$ in the third equation because $p(y)$ does not depend on $i$. In other words, to find the optimum detector, we simply find the hypothesis $i$ whose product $\pi_i p(y|H = i)$ is larger than all of the other hypotheses.

Still considering uniform costs, we can derive this same detector a bit more directly using Bayes' rule. Since the Bayes risk is simply the probability that the detector makes a mistake, all we need to do is to find out which hypothesis is most probable given the observation $y$. This leads to a detector that we can express very, very simply:

$$\delta(y) = \arg\max_i p(H = i|y). \qquad (3.91)$$

Using Bayes rule, we can write this in terms of the likelihood functions and the priors:

$$\delta(y) = \arg\max_i \frac{p(y|H = i)\pi_i}{p(y)}. \qquad (3.92)$$

As before, the marginal $p(y)$ is independent of $i$, so we can eliminate it from the maximization problem. Therefore, we get

$$\delta(y) = \arg\max_i p(y|H = i)\pi_i, \qquad (3.93)$$

just as before. This rule is simple to implement conceptually, and as long as the number of hypotheses is not too large it is easy to implement in practice, too.

**Example 3.12: Multiple Object Detection.** *Let's look at a varation on our favorite object detection problem. Consider three hypotheses: $H_0$, the hypothesis that no object is present, $H_1$, the hypothesis that* one *type of object is present, and $H_2$, the hypothesis that* another *type of object is present. The two types of objects may, for example, differ in size, resulting in a different observation signature. We model this with the following likelihood functions:*

$$
\begin{aligned}
H_0 &: Y \sim \mathcal{N}(0, \sigma^2) && (3.94)\\
H_1 &: Y \sim \mathcal{N}(a, \sigma^2) && (3.95)\\
H_2 &: Y \sim \mathcal{N}(b, \sigma^2), && (3.96)
\end{aligned}
$$

*where $0 < a < b$. Also suppose uniform priors and costs. Then, the optimum detector boils down to three tests. The detector chooses $H_0$ when*

$$
\begin{aligned}
&\exp(-y^2/(2\sigma^2)) \geq \exp(-(y-a)^2/(2\sigma^2)), \exp(-(y-b)^2/(2\sigma^2)) && (3.97)\\
\Rightarrow &\exp(-y^2/(2\sigma^2)) \geq \exp(-(y-a)^2/(2\sigma^2)) && (3.98)\\
\Rightarrow &\, y \leq \frac{a}{2}. && (3.99)
\end{aligned}
$$

*We can neglect the condition on* $\exp(-(y-b)^2/(2\sigma^2))$ *because whenever* $\exp(-y^2/(2\sigma^2)) \geq \exp(-(y-a)^2/(2\sigma^2))$*,* $b > a$ *implies* $\exp(-(y-a)^2/(2\sigma^2)) \geq \exp(-(y-b)^2/(2\sigma^2))$*. Similarly, the detector chooses* $H_2$ *when*

$$\exp(-(y-b)^2/(2\sigma^2)) \geq \exp(-(y-a)^2/(2\sigma^2)), \exp(-y^2/(2\sigma^2)) \tag{3.100}$$

$$\Rightarrow \exp(-(y-b)^2/(2\sigma^2)) \geq \exp(-(y-a)^2/(2\sigma^2)) \tag{3.101}$$

$$\Rightarrow y \geq \frac{a+b}{2}. \tag{3.102}$$

*This leads to the optimum detector*

$$\delta(y) = \begin{cases} 0, & y \leq \frac{a}{2} \\ 1, & \frac{a}{2} < y < \frac{a+b}{2} \\ 2, & y \geq \frac{a+b}{2} \end{cases}. \tag{3.103}$$

*In this case, the optimum detector combines two binary detectors:* $H_0$ *vs.* $H_1$ *and* $H_1$ *vs.* $H_2$*. This is not true in general. Detection regions in multiple-hypothesis testing can be rather complicated. For example, what happens if the priors are not uniform?*

We can apply these ideas to more complicated problems.

**Example 3.13: Different Means, Isotropic Covariances.** *We'll suppose that the different hypotheses are associated with multivariate Gaussian likelihoods with different means, but identical covariances* $\sigma_2\mathbf{I}$*:*

$$H_i : Y \sim \mathcal{N}(\mu_i, \sigma^2\mathbf{I}), \tag{3.104}$$

*where the priors* $\pi_i$ *are left arbitrary. We can also use the* log-likelihood *functions in computing the optimum detector, which leads to*

$$\delta(\mathbf{y}) = \arg\max_i \log p(y|H=i) + \log \pi_i \tag{3.105}$$

$$= \arg\max_i \|\mathbf{y} - \mu_i\|^2 + \log \pi_i, \tag{3.106}$$

*where the rest of the terms in the log-likelihood function can be eliminated from the detector because they are the same across all hypotheses. In other words, the Bayes-optimum detector just finds the* $\mu_i$ *closest to the received signal* $\mathbf{y}$*, adjusted by the strength of the prior. In the case of uniform priors, the optimum detector simply minimizes the Euclidean distance to* $\mathbf{y}$*.*

Generalizing this slightly, we can examine a scenario that's quite relevant to digital communications.

**Example 3.14: Digital Communications: Phase Shift Keying.** *You probably know that digital wireless communications systems modulate bits into sinusoidal functions at the transmit frequency; here we'll dig into the details. A very popular approach is called* **phase shift keying** *(PSK), in which the bits are encoded in the* phase *of the sinusoid transmitted. This is accomplished by defining a* **constellation** *of phases, each corresponding to a different sequence of bits. For concreteness, we'll focus on* **quaternary phase shift keying** *(QPSK), where two bits are encoded into the four possible phases* $0, \pi/2, \pi,$ *and* $3\pi/4$*.*

*Suppose uniform priors* $\pi_i = 1/4$*. We suppose a carrier signal at (discrete-time) frequency* $\omega \in (0, 2\pi)$*, and we suppose that the bits are transmitted over* $n$ *(discrete-time) samples. Let* $T$ *denote* *the number of periods of the carrier sinusoid that the samples span.*[5] *We also suppose that the recieve signal is corrupted by i.i.d. Gaussian noise of variance $\sigma^2$ per element. Then, the signal received can be written as*

$$y_k = a\cos(\omega k) + b\sin(\omega k) + n_k, \tag{3.107}$$

*for $1 \le k \le n$, where $n_k \sim \mathcal{N}(0, \sigma^2)$, and where the pair $(a, b)$ is equal to $(1, 0)$ for $M = 0$, $(0, 1)$ for $M = 1$, $(-1, 0)$ for $M = 2$, and $(0, -1)$ for $M = 3$. You should convince yourself (via Euler's identity) that these choices correspond to cosines with the four phase shifts specified above.*

*This scenario corresponds to the case above of Gaussian signals with different means but same covariances, and we* can *design a detector that simply looks for the $n$-dimensional mean vector closest to* **y** *in the sense of Euclidean norm. But the structure of the signal allows us to do something even more clever. Specifically, define*

$$\hat{a} = \frac{2}{n}\sum_{k=1}^{n} y_k \cos(\omega k) = a + N_a \tag{3.108}$$

$$\hat{b} = \frac{2}{n}\sum_{k=1}^{n} y_k \sin(\omega k) = b + N_b, \tag{3.109}$$

*where it is straightforward to show that $N_a, N_b \sim \mathcal{N}(0, \sigma^2/n)$, and where the relationship between $\hat{a}$ and $\hat{b}$ is due to the fact that the sine and cosine functions are* orthogonal. *From this, we can carry out the detection in two dimensions, and it's straightforward to see that we choose the hypothesis corresponding to the $(a, b)$ closest in Euclidean distance to $(\hat{a}, \hat{b})$. Furthermore, since the variance on the effective noise terms $N_a, N_b$ decreases as $1/n$, the performance of the detector improves the larger $n$ is. That is, the slower the data rate (or the more samples we use to encode a pair of bits), the more reliable the communications.*

*This form of detector, called the* **matched filter**, *enjoys widespread use throughout signal processing and communications. As we receive the sequence $y_k$, we multiply it element-by-element by the sine and cosine signals. We can think these as the coefficients of a simple filter. Even more formally, the matched filter computes the inner product between the vector* **y** *and the sine and cosine, computing the* projection *onto the signal space elements defined by sine and cosine functions. If the hypotheses correspond to different signals in noise than sinusoids, one can define a matched filter that corresponds to those signals.*

### 3.2.2 Maximum-likelihood Hypothesis Testing

What are we to do if there are no priors available? We cannot employ the Neyman-Pearson criteria; we have more than two hypotheses, so what does a "false alarm" or a "missed detection" mean? A minimax criterion is possible in principle, but in practice constructing a minimax test would be an arduous task. Instead, the most popular approach is to employ **maximum-likelihood detection**, in which we simply choose the hypothesis that maximizes the likelihood function:

$$\delta(y) = \arg\max_i p(y|H = i). \tag{3.110}$$

It's important to stress that this test does *not* choose the most probable hypothesis. The likelihood function $p(y|H = i)$ is not a probability distribution over $H$, and without a prior we cannot compute

[5] For simplicity, we assume that the $n$ is an integer multiple of the fundamental period $2\pi/\omega$ of the carrier signal.

a probability distribution over $H$. Instead, the maximum-likelihood detector is a heuristic. It corresponds to "guessing" that the prior is uniform over the $M$ hypotheses and taking the costs to be uniform. This assumption holds in a few cases—for example, in digital communications, symbols usually are uniformly distributed over the constellation. But even when the assumption is not warranted, the maximum-likelihood detector is extremely popular in practice.

## 3.3 Composite Hypothesis Testing

Let's consider a different scenario, which is a bit closer to the estimation framework we'll examine in later chapters. Let $\Lambda$ be a **parameter space**. Usually we will think of $\Lambda$ as a continuous set, usually a subset of Euclidean space, but in principle it is arbitrary. Similar to the hypotheses $H_0$ and $H_1$, each $\theta \in \Lambda$ describes a possible state of nature. For each $\theta \in \Lambda$, the likelihood function $p(y|\theta)$ describes the distribution of $Y$ over the corresponding state of nature.

For example, $\theta$ could represent the turnout of an election, and $p(y|\theta) = \mathcal{N}(\theta, \sigma^2)$ might be the distribution of the exit polls, the mean of which is the true turnout but which is corrupted by measurement noise. Similarly, $\theta$ might be the velocity or altitude of an aircraft, which we measure subject to noise. In later chapters, we will discuss *estimating* $\theta$ from noisy measurements. But in this section we are most emphatically *not* interested in knowing $\theta$ directly.

Instead, composite hypothesis testing involves partitioning the set $\Lambda$ into two sets $\Lambda_0$ and $\Lambda_1$, which correspond to null and alternative hypotheses $H_0$ and $H_1$. The set $\Lambda_0$ corresponds to "ordinary" states of nature—voter turnouts close to historical averages or aircraft speeds near a nominal value, perhaps—whereas $\Lambda_1$ corresponds to anomalous states of nature. Rather than estimate $\theta$ exactly, the objective is only to determine if it is in the ordinary set or in the anomalous set.

As with ordinary hypothesis testing, we can carry out composite hypothesis testing using the Bayes criteria. The Neyman-Pearson criteria, on the other hand, is possible only under certain rather stringent circumstances.

### 3.3.1 Bayes Composite Hypothesis Testing

For the Bayesian framework, we need a prior $\pi(\theta)$ over the parameter space. This requirement is much more burdensome than in ordinary hypothesis testing. The latter scenario requires that we know only a binary distribution; here we need a distribution over all of $\Lambda$. If the existence of a prior is controversial for simple hypothesis testing, it is all the more so for composite hypothesis testing!

As before, we also need the cost assignments $C_{00}$, $C_{10}$, $C_{01}$, and $C_{10}$ that describe the relative cost of the possible outcomes. Again we make the restriction that $C_{00} < C_{10}$ and $C_{11} < C_{01}$; correct detection is always less costly than incorrect detection. The resulting cost function, which we want to minimize, is

$$r(\delta) = C_{00}\Pr(H_0, \delta(Y) = 0) + C_{10}\Pr(H_0, \delta(Y) = 1) + C_{01}\Pr(H_1, \delta(Y) = 0) + C_{11}\Pr(H_1, \delta(Y) = 1) \tag{3.111}$$

$$= C_{00}\int_{\Gamma_0} p(H_0|y)dy + C_{10}\int_{\Gamma_1} p(H_0|y)dy + C_{01}\int_{\Gamma_0} p(H_1|y)dy + C_{11}\int_{\Gamma_1} p(H_1|y)dy \tag{3.112}$$

$$= \int_{\Gamma_0} (C_{00}p(H_0|y) + C_{01}p(H_1|y) + \int_{\Gamma_1} (C_{10}p(H_0|y)dy + C_{11}p(H_1|y)dy. \tag{3.113}$$

Using the same argument as in the case of simple hypothesis testing, the optimum detector uses

$$\Gamma_0 = \left\{ y : \frac{p(H_1|y)}{p(H_0|y)} < \frac{C_{10} - C_{00}}{C_{01} - C_{11}} \right\} \tag{3.114}$$

$$\Gamma_1 = \left\{ y : \frac{p(H_1|y)}{p(H_0|y)} \geq \frac{C_{10} - C_{00}}{C_{01} - C_{11}} \right\}. \tag{3.115}$$

This is similar in form to the LRT test from the simple case, but with the crucial difference that it involves the ratio of *probabilities*, not likelihoods. We need to compute the posterior, which is often challenging.

With a little bit of work, we can rework the problem so that the optimum detector is of the form of an LRT. First, we compute the prior probability of the null and alternative hypotheses by integrating over the prior:

$$\pi_0 = \int_{\Lambda_0} p(\theta)d\theta \tag{3.116}$$

$$\pi_1 = 1 - \pi_0 = \int_{\Lambda_1} p(\theta)d\theta. \tag{3.117}$$

Second, we compute the likelihood function, which requires a bit more work. Using Bayes' rule, we compute

$$p(y|H_0) = \frac{p(H_0|y)p(y)}{p(H_0)} \tag{3.118}$$

$$= \frac{1}{\pi_0}\int_{\Lambda_0} p(\theta|y)p(y)d\theta. \tag{3.119}$$

Of course, we don't *know* $p(\theta|y)$. If we did, we could just carry out the hypothesis test given above in terms of the posterior probabilities. To get this in terms of the likelihood function we know, $p(y|\theta)$, we apply Bayes' rule again:

$$p(y|H_0) = \frac{1}{\pi_0}\int_{\Lambda_0} \frac{p(y|\theta)p(\theta)}{p(y)}p(y)d\theta \tag{3.120}$$

$$= \frac{1}{\pi_0}\int_{\Lambda_0} p(y|\theta)p(\theta)d\theta. \tag{3.121}$$

In other words, the likelihood function for hypothesis $H_0$ is proportional to the "natural" likelihood function $p(y|\theta)$, averaged over the prior over the region $\Lambda_0$. It's straightforward to carry out the same derivation for $H_1$, which gives us a similar result:

$$p(y|H_1) = \frac{1}{\pi_1}\int_{\Lambda_1} p(y|\theta)p(\theta)d\theta. \tag{3.122}$$

As long as we can compute these integrals, we can compute all of the elements of the simple Bayes-optimum detector. In this case, the optimum detector is described by the decision regions

$$\delta(y) = \begin{cases} 0, & \text{if } \frac{p(y|H=1)}{p(y|H=0)} < \frac{(C_{10}-C_{00})\pi_0}{(C_{01}-C_{11})\pi_1} \\ 1, & \text{if } \frac{p(y|H=1)}{p(y|H=0} \geq \frac{(C_{10}-C_{00})\pi_0}{(C_{01}-C_{11})\pi_1} \end{cases}, \tag{3.123}$$

just as in the simple case.

Observe the essential nature of the prior in composite hypothesis testing. If we do not know $p(\theta)$, not only can we not compute the priors $\pi_0$ and $\pi_1$, but we also cannot compute the likelihood functions $p(y|H_0)$ and $p(y|H_1)$! This is a substantial hurdle if we want, say, to construct a minimax or Neyman-Pearson test for circumstances in which the prior is unknown. It turns out that we do not strictly require knowledge of $p(\theta)$ to compute the likelihoods. It is sufficient to know the ratios $p(\theta)/\pi_0$ and $p(\theta)/\pi_1$ over the regions $\Lambda_0$ and $\Lambda_1$, respectively.

This is small comfort. It is difficult to imagine a scenario in which we have access to—or can confidently assume—a normalized version of the prior but cannot access or assume the prior itself. In order to develop a Neyman-Pearson theory for composite hypotheses, we will need to try something different.

### 3.3.2 Neyman-Pearson Composite Hypothesis Testing

In "simple" hypothesis testing, we defined the false alarm probability $P_F(\delta)$ and detection probability $P_D(\delta)$, and we found the detector that maximized the detection probability while maintaining a fixed false-alarm probability. We will follow the same intuition, but the lack of an available prior distribution will force us to modify both the Neyman-Pearson criterion and the resulting detector.

The standard definition of $P_F(\delta)$ requires the likelihood function of the null hypothesis. Because we do not have the prior $p(\theta)$, we cannot compute the likelihood function $p(y|H=0)$. Instead, we define the size of the test as the *worst-case* false-alarm probability over the parameters associated with the null hypothesis:

$$\bar{P}_F(\delta) = \sup_{\theta\in\Theta_0} \Pr(\delta(y)=1|\theta) \tag{3.124}$$

$$= \sup_{\theta\in\Theta_0} \int_{\Gamma_1} p(y|\theta)dy. \tag{3.125}$$

This definition of the size considers the false alarm probability of each component $\theta \in \Theta_0$ of the null hypothesis individually, and then returns the probability of the worst-case component. We do not need access to a prior $p(\theta)$ in order to compute it; the original likelihood function $p(y|\theta)$ is sufficient.

Following the Neyman-Pearson criterion for simple hypotheses, we would like to maximize the detection probability for a fixed size $\bar{P}_F(\delta) = \alpha$. Again, not having access to $p(y|H=1)$ prevents us from realizing this criterion directly. Instead, we will look for a test that is most powerful for every component $\theta \in \Theta_1$ of the alternative hypothesis. Define the power of the detector $\delta$ pointwise as

$$P_D(\delta,\theta) = \int_{\Gamma_1} p(y|\theta)dy, \tag{3.126}$$

for every $\theta \in \Theta_1$. Then, we seek the detector that maximizes the power *uniformly.*

**Definition 3.7:.** *A detector* $\delta(y)$ *is* **uniformly most powerful of size** $\alpha$ *for testing* $H_0 : \theta \in \Theta_0$ *against* $H_1 : \theta \in \Theta_1$ *if*

$$\bar{P}_F(\delta) \leq \alpha, \tag{3.127}$$

*and*

$$P_D(\delta, \theta) \geq P_D(\delta', \theta), \tag{3.128}$$

*for every* $\theta \in \Theta_1$ *and for every* $\delta'$ *with* $\bar{P}_F(\delta') \leq \alpha$.

The UMP criteria is rather stringent, and the UMP test does not always exist. This is by contrast to simple hypothesis testing, in which the Neyman-Pearson lemma guarantees the existence of an optimum test in the form of a likelihood ratio test. Indeed, for any $\theta \in \Theta_1$, the Neyman-Pearson lemma guarantees the existance of a test of size $\alpha$ that maximizes $P_D(\delta, \theta)$, but there is no reason that the same test will maximize the power for a different $\theta' \in \Theta_1$. This suggests a sufficient condition for the existence of a UMP test: if the Neyman-Pearson for every simple alternative hypothesis test $\theta \in \Theta_1$ is the same, then that detector is a UMP test. A stringent requirement indeed!

When the UMP test is not available, an alternative choice is to consider the *generalized* likelihood ratio

$$\Lambda(y) = \frac{\sup_{\theta \in \Theta_0} p(y|\theta)}{\sup_{\theta \in \Theta} p(y|\theta)}, \tag{3.129}$$

and to choose the test threshold that ensures $\bar{P}_D(\delta) = \alpha$. This test does not come with the optimality guarantees of the UMP, but it always exists.

# Chapter 4

# Elements of Parameter Estimation

In detection, we wanted to determine which of several hypotheses was true, which we did by performing a likelihood ratio test (in the case of binary hypotheses), or some generalization thereof (in the case of multiple or composite hypotheses.) In this chapter, we will consider the more challenging problem of *estimating* a continuous parameter from observations. For example, we may want to estimate via radar the range to an object, the frequency of a carrier wave in wireless communications, or the average value of a biased coin flip. In the language of statistics and machine learning, we will usually think of estimation as fitting a *model* to the data we collect. Because there are (uncountably) infinite possibilities for the parameter, there simply is no notion of a probability of error—virtually every estimate is incorrect. Instead, we must decide how to quantify estimation error—what makes an estimator "good" or "bad"? These choices will result in rather different estimators.

First, we will consider *non-random* estimation. This is roughly analogous to the Neyman-Pearson framework, where we do not have access to a prior distribution. Indeed, as the name suggests, we will suppose that the parameter to be estimated is not a random variable at all. In non-random estimation, we consider two main measures of estimate quality: the *bias* and the *variance.* If an estimator has low bias, then the expected value of the estimate is close to the true parameter value, and if it has low variance, the expected *squared* error is close to zero. In non-random estimation, we will spend most of our effort seeking the unbiased estimator that has the minimum variance—the so-called *minimum variance unbiased* (MVUB) estimator. Then, we will investigate the properties of the MVUB estimator. We will derive a powerful bound, called the *Cramer-Rao* bound, on the variance of the MVUB estimator. Finally, we will examine the simple *maximum-likelihood* (ML) estimator. We'll show that even though the ML estimator is not usually the MVUB, in the asymptote of many observations the ML estimator approaches an unbiased estimator whose variance matches the Cramer-Rao bound. For this reason, the ML estimator is tremendously popular in practice.

Then, we will study Bayesian estimation, where we suppose that we have access to a prior distribution over the parameter space. In the Bayesian framework, we seek to minimize the *average* estimation error, which we will measure via three different error functions. These three functions will give rise to three different Bayes estimators. We will find that different choices of prior will lead to rather different estimation schemes, especially in the case when the signal dimensionality is high.

We also will introduce the basics of machine learning, particularly **supervised** and **unsupervised**

**learning**. In both the non-random and Bayesian settings, we can design classification, clustering, and feature extraction algorithms that are widely used in practice.

## 4.1 Non-random Estimation

In parameter estimation, we suppose that we observe a signal $Y \in \Gamma$ that is drawn from a distribution that is *parameterized by* a parameter $\theta \in \Lambda$ according to the **likelihood function**

$$p(y|\theta), y \in \Gamma, \theta \in \Lambda. \tag{4.1}$$

In other words, the observed signal $Y$ comes from one of a *family* of distributions, indexed by the parameter $\theta$. We know the family, but we do not know the parameter, and we want to estimate $\theta$ from $Y$. Thus, we define an **estimator**

$$\hat{\theta}(y) : \Gamma \to \Lambda. \tag{4.2}$$

Often the received signal will consist of $n$ i.i.d. samples drawn from $p(y|\theta)$. In this case, we let $Y^n$ denote the signal, and the likelihood function is

$$p(y^n|\theta) = \prod_{i=1}^{n} p(y_i|\theta). \tag{4.3}$$

An obvious intuition, which we'll establish rigorously, is that we can build a better estimator if we have more samples!

Let's make this concrete with a few examples.

**Example 4.1: A Biased Coin.** *Let's consider the estimation of the bias of a coin from $n$ flips. We model this with $\Gamma = \{0, 1\}$ to refer to the outcome of a coin, and take the Bernoulli distribution as the likelihood function*

$$p(y|\theta) = \begin{cases} 1-\theta, & y = 0 \\ \theta, & y = 1 \end{cases}, \tag{4.4}$$

*where $\theta$ takes the role of $p$ in the definition given in Chapter 2. A very reasonable estimator is the sample mean:*

$$\hat{\theta}(y^n) = \frac{1}{n}\sum_{i=1}^{n} y_i, \tag{4.5}$$

*which just counts how many times $Y_i = 1$. We will see later that this estimator has nice properites.*

An another:

**Example 4.2: Gaussian Distribution with Unknown Mean.** *Suppose that $\Gamma = \mathbb{R}^d$, and that the signal $\mathbf{y}$ is drawn from a Gaussian with known covariance but unknown mean, i.e.*

$$p(y|\theta) = \mathcal{N}(\mu, \Sigma). \tag{4.6}$$

*Then, the parameter is $\theta = \mu$, and the parameter space is $\Lambda = \mathbb{R}^d$. A sensible estimator here is also the sample mean*

$$\hat{\mu}(\mathbf{y}^n) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i, \tag{4.7}$$

*where here the estimate is a vector in $\mathbb{R}^d$.*

Finally, we may want to consider the case in which both the mean and covariance are unknown:

**Example 4.3: Gaussian Distribution with Unknown Mean and Covariance.** *Suppose that* $\Gamma = \mathbb{R}^d$*, and that the signal* $\mathbf{y}$ *is drawn from a Gaussian with unknown mean and covariance, i.e.*

$$p(\mathbf{y}|\theta) = \mathcal{N}(\mu, \Sigma). \tag{4.8}$$

*Then, the parameter is* $\theta = (\mu, \Sigma)$*, and the parameter space is* $\Lambda = \mathbb{R}^d \times \mathbb{R}^{d\times d}$*. We can estimate the mean via the sample mean as before,*

$$\hat{\mu}(\mathbf{y}^n) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i, \tag{4.9}$$

*and the covariance via the sample covariance*

$$\hat{\Sigma}(\mathbf{y}^n) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\mu})(\mathbf{y}_i - \hat{\mu})^T. \tag{4.10}$$

*It turns out that the sample covariance estimator is not always a good idea, for reasons that we will see shortly.*

The likelihood function and estimator are analogous to the likelihood and detector in hypothesis testing. In hypothesis testing, we aimed to infer which of finitely many hypotheses $H_0, H_1, \ldots$ is true from the observation $Y$. We did this by choosing the detector that gives the optimum tradeoff between false alarm and detection probabilities. In estimation, we want to infer the value of a parameter $\theta \in \Lambda$. The parameter $\theta$ is usually a continuous random variable, and it may also be a vector. The observation $Y$ will also usually (but not always!) be a continuous random variable (or vector), in which case $p(y|\theta)$ is a (perhaps multivariate) density function. In so-called "non-random" estimation, we will not treat the parameter $\theta$ as a random variable, but merely as a state of nature that we want to estimate. The theory built up around this approach is at the heart of much of classical and modern statistics.

The challenge in estimation theory is to define meaningful metrics of estimation quality. We will be primarily concerned with the *bias* and *variance* of an estimator.

**Definition 4.1:.** *We say that an estimator* $\hat{\theta}(y)$ *is* unbiased *if*

$$E[\hat{\theta}(y)] = \theta, \forall \theta \in \Lambda, \tag{4.11}$$

*where the expectation is computed according to* $p(y|\theta)$*.*

Bias is a property that holds uniformly over values of $\theta$, a crucial fact because $\theta$ is not a random variable. Think of it this way: pick an arbitrary $\theta_0 \in \Lambda$. Then, $Y \sim p(y|\theta_0)$, and the estimator $\hat{\theta}(y)$ is a random variable whose distribution is determined by $p(y|\theta_0)$. The estimator $\hat{\theta}(y)$ is said to be unbiased only if, for *every single* $\theta_0$, the expected value of $\hat{\theta}(y)$ is equal to $\theta_0$.

**Example 4.4: Gaussian Distribution with Unknown Mean.** *We consider the case of* $n$ *i.i.d. samples from a Gaussian distribution with known variance and unknown mean:*

$$p(y|\theta) = \prod_{i=1}^{n} \mathcal{N}(\mu, \Sigma). \tag{4.12}$$

*We saw before that an obvious estimator is the sample mean:*

$$\hat{\theta}(\mathbf{y}^n) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i. \tag{4.13}$$

*It turns out that this simple estimator is indeed unbiased:*

$$E[\hat{\theta}(y)] = E\left[\frac{1}{n}\sum_{i=1}^{n}Y_i\right] = \frac{1}{n}\sum_{i=1}^{n}E[Y_i] = \frac{1}{n}\sum_{i=1}^{n}\theta = \theta. \tag{4.14}$$

The case of unknown (co)variance is more complicated.

**Example 4.5: Gaussian Distribution with Unknown Mean and Variance.** *We consider the scalar version of the example above, $n$ i.i.d. samples from a Gaussian distribution where the mean and variance are both unknown. Let $\theta = (\mu, \sigma^2)$*

$$p(y^n|\theta) = \prod_{i=1}^{n}\mathcal{N}(\mu, \sigma^2). \tag{4.15}$$

*Here, the obvious estimators are the sample mean and the sample variance. Let $\hat{\theta}(y) = (\hat{\mu}(y), \hat{\sigma^2}(y))$, where*

$$\hat{\mu}(y^n) = \frac{1}{n}\sum_{i=1}^{n}y_i \tag{4.16}$$

$$\hat{\sigma^2}(y^n) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}(y))^2. \tag{4.17}$$

*From the previous example we know that $\hat{\mu}$ is unbiased. What about $\hat{\sigma^2}$?*

$$\begin{aligned}
E[\hat{\sigma^2}] &= \frac{1}{n}\sum_{i=1}^{n}E\left[\left(Y_i - \frac{1}{n}\sum_{j=1}^{n}Y_j\right)^2\right] && (4.18)\\
&= \frac{1}{n}\sum_{i=1}^{n}E\left[Y_i^2 - \frac{2Y_i}{n}\sum_{j=1}^{n}Y_j + \left(\frac{1}{n}\sum_{j=1}^{n}Y_j\right)^2\right] && (4.19)\\
&= E[Y_i^2] - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}E[Y_iY_j] && (4.20)\\
&= E[Y_i^2] - \frac{1}{n^2}(n(n-1)\mu^2 - nE[Y_i^2]) && (4.21)\\
&= \frac{n-1}{n}(E[Y_i^2] - \mu^2) = \sigma^2\frac{n-1}{n}. && (4.22)
\end{aligned}$$

*The sample variance is a biased estimate of $\sigma^2$. A simple solution is to multiply the estimate by $n/(n-1)$, which corrects the bias. However, also note that the bias becomes small as $n$ increases. If there are enough data samples, the estimator is "nearly" unbiased. Both the "nearly-unbiased" sample variance and the corrected unbiased estimator are used widely in practice.*

*In the multivariate case, where* $p(\mathbf{y}^n|\theta) = \prod_{i=1}^n \mathcal{N}(\mu, \Sigma)$, *one can follow similar steps to establish that the* **sample covariance** *estimator* $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (\mathbf{y}_i - \hat{\mu})(\mathbf{y}_i - \hat{\mu})^T$ *is also biased, with*

$$E[\hat{\Sigma}] = \frac{n-1}{n}\Sigma. \tag{4.23}$$

*A similar correction by a factor of* $n/(n-1)$ *gives an unbiased estimate.*

We can think of an unbiased estimator as the true parameter $\theta$ plus additive noise—the distribution of which is unknown, but which is known to be zero mean. That is, $\hat{\theta} = \theta + \mathbf{n}$, where $\mathbf{n}$ is a zero mean random variable/vector that describes the estimation error. The (co)variance of this estimation error quantifies how good of an estimate $\hat{\theta}$ is.

Indeed, the second measure of estimation quality is the covariance.

**Definition 4.2:.** *For* $\Lambda = \mathbb{R}$, *the* **(co)variance** *of an estimator is*

$$\mathrm{var}(\hat{\theta}) = E[(\hat{\theta}(y) - E[\hat{\theta}(y)])(\hat{\theta}(y) - E[\hat{\theta}(y)])^T]. \tag{4.24}$$

Note that this is the covariance of the estimator itself, *not* the covariance of the estimation error. These quantities are only the same when $\hat{\theta}(y)$ is unbiased; then $E[\hat{\theta}(y)] = \theta$, and $\mathrm{var}(\hat{\theta})$ can be thought of as the covariance of the additive noise $\mathbf{n}$ above.

We naturally prefer small estimation covariance, but only to a point. Indeed, there is an inherent trade-off between bias and variance: The lower the bias of $\hat{\theta}$, the higher the variance, and vice versa. To see this, observe that it is *always* possible to choose an estimator with zero variance. If we take $\hat{\theta}(y) = 0$, regardless of $y$, then the estimator is a constant and has no variance. This is a rather bad estimator, however, because it has high bias. Instead, let's look at the variance of the unbiased estimator of the mean that we found above.

**Example 4.6: Gaussian Distribution with Unknown Mean.** *Returning to the setting of* $n$ *i.i.d. scalar Gaussians with known variance and unknown mean, an unbiased estimator is*

$$\hat{\mu}(y^n) = \frac{1}{n}\sum_{i=1}^n y_i. \tag{4.25}$$

*The variance of this estimator is*

$$\begin{aligned}
E[(\hat{\mu}(Y^n) - \mu)^2] &= E\left[\left(\frac{1}{n}\sum_{i=1}^n Y_i - \mu\right)^2\right] && (4.26)\\
&= E\left[\frac{1}{n^2}\sum_{i=1}^n\sum_{i=1}^n Y_iY_j - \frac{2}{n}\sum_{i=1}^n Y_i\mu + \mu^2\right] && (4.27)\\
&= \frac{1}{n^2}\sum_{i=1}^n\sum_{i=1}^n E[Y_iY_j] - \mu^2 && (4.28)\\
&= \frac{n-1}{n}\mu^2 + 1/nE[Y_i^2] - \mu^2 && (4.29)\\
&= 1/nE[Y_i^2] - \mu^2/n = \sigma^2/n. && (4.30)
\end{aligned}$$

*The variance of the estimate is inversely proportional to the number of samples. Similarly, in the multivariate case, we have for the sample mean estimator*

$$\mathrm{var}(\hat{\mu}) = \frac{1}{n}\Sigma \tag{4.31}$$

But is this the *best* estimator we can construct? We could always reduce the variance by changing the $\frac{1}{n}$ factor in the sample mean to some other value, say $\frac{1}{n+c}$. This estimator has variance $\frac{\sigma^2}{/}n + c$, but is clearly biased. When we study Bayesian estimation we will see that such a biased estimate is ideal when we have access to a prior distribution $p(\theta)$; in the current setting we only can say that it allows us to trade freely between bias and variance.

To answer this question in a specific way, we will restrict our attention to unbiased estimators, and we will look for the estimator with the smallest (co)variance.

### 4.1.1 Minimum-variance Unbiased Estimators

**Definition 4.3:.** *We call a scalar estimator the* **minimum variance unbiased (MVUE) estimator** $\hat{\theta}_{MVUE}(y)$ *if*

$$E[\hat{\theta}_{MVUE}(y)] = \theta, \tag{4.32}$$

*and there is no other unbiased estimator* $\hat{\theta}^*$ *such that*

$$E[(\hat{\theta}^*(y) - \theta)^2] < E[(\hat{\theta}_{MVUE}(y) - \theta)^2], \tag{4.33}$$

*for all* $\theta$.

In other words, an estimator is a MVUE if $\hat{\theta} = \theta + n$, where $n$ is zero-mean noise with the smallest possible variance. As with bias, the MVUB has a *uniform* property: it must not only be unbiased, but it must have the minimum variance for *every* $\theta \in \Lambda$. As a result, the MVUB may not exist. If there are multiple unbiased estimators, and one estimator has the minimum variance for some subset of $\Lambda$ but another has the minimum variance for another subset, the MVUE does not exist. We will look at a few seemingly simple estimation cases that suffer from this condition.

Even when the MVUB does exist, finding it is a difficult task, and there is no single procedure that one can follow. Instead, we will usually employ a "guess and check" approach: use good judgment to propose a "reasonable" unbiased estimator, and check to see if it is MVUE.

In the examples above, the "reasonable" estimators were averages of *functions* of the data chosen to mimic the thing we're trying to estimate: the sample mean, the sample covariance, etc., compute averages of functions that *look like* the mean and the covariance. It is therefore not surprising that they produce unbiased estimates of the mean and covariance. We'll see in a bit that these estimators are also MVUE.

Can we generalize this process beyond the Gaussian examples we've looked at so far? Indeed we can, and to do so we'll need the concept of a **sufficient statistic**. A **statistic** $T(Y)$ of the observation $Y$ is some function that "pre-processes" the data prior to estimation. For example, in estimating the mean of a Gaussian distribution, the estimator was a function of the sample mean $\bar{Y} = 1/n \sum_{i=1}^{n} Y_i$. This statistic is easier to work with than the full data set, and we didn't lose any information that we needed in order to estimate $\theta$. In that sense, the statistic was *sufficient* for estimating the mean. We formalize this idea in the following definition.

**Definition 4.4:.** *A function* $T(y)$ *is said to be a* **sufficient statistic** *for* $\theta$ *if the distribution of* $Y$ *conditioned on* $T(Y)$ *does not depend on* $\theta$*, i.e.* $p(Y|T(Y), \theta)$ *is not a function of* $\theta$.

To verify that a statistic $T(Y)$ is sufficient, we need to verify that the conditional distribution $p(Y|T(Y), \theta)$, which still is *parameterized* by $\theta$, indeed does not depend on $\theta$.

**Example 4.7: Gaussian Sample Mean.** *We again look at the case of $n$ i.i.d. Gaussian random variables with known variance and unknown mean $\theta$. Consider the statistic*

$$T(Y) = \sum_{i=1}^{n} Y_i. \tag{4.34}$$

*Is it sufficient?*

*To find out, we need to calculate the conditional distribution of $Y$, which according to the definition of conditional probability is*

$$p(y|T(y) = T_0, \theta) = \frac{p(y, T(y) = T_0|\theta)}{p(T(y) = T_0|\theta)}. \tag{4.35}$$

*The joint probability in the numerator simply factors as $p(y, T(y) = T_0|\theta) = p(|;\theta)\delta(T(y) - T_0)$, where $\delta(\cdot)$ is the Dirac delta function. In other words, the joint probability is the likelihood function, restricted to the case where $T(y) = T_0$. Thus, can rewrite the numerator as*

$$\begin{aligned} p(y, T(y) = T_0|\theta) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \theta)^2\right) && (4.36)\\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n} y_i^2 - 2\theta T(y) - n\theta^2\right)\right)\delta(T(y) - T_0) && (4.37)\\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n} y_i^2 - 2\theta T_0 - n\theta^2\right)\right)\delta(T(y) - T_0). && (4.38) \end{aligned}$$

*On the other hand, $T(y)$ is simply a Gaussian random variable with mean $n\theta$ and variance $n\sigma^2$. The conditional distribution is therefore*

$$\begin{aligned} p(y, T(y) = T_0|\theta) &= \frac{\frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2\right)\exp\left(-\frac{1}{2\sigma^2}(-2\theta T_0 - n\theta^2)\right)\delta(T(y) - T_0)}{\frac{1}{(2\pi n\sigma^2)^{n/2}}\exp\left(-\frac{1}{2n\sigma^2}(T_0 - n\theta)^2\right)} && (4.39)\\ &= \frac{\sqrt{n}}{(2\pi\sigma^2)^{(n-1)/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2\right)\exp\left(\frac{T_0^2}{2n\sigma^2}\right)\delta(T(y) - T_0), && (4.40) \end{aligned}$$

*which does not depend on $\theta$. This means that the sum of the observation is a sufficient statistic. By extension, the sample average is also a sufficient statistic. In fact, any scale multiple (or invertible function) of $T(Y)$ is sufficient for $\theta$.*

Let's try out a non-Gaussian example.

**Example 4.8: Bernoulli Samples.** *Suppose we get access to $n$ samples $Y_i \sim \mathcal{B}(\theta)$ drawn from a Bernoulli distribution with parameter $p$, giving us the likelihood*

$$p(y|\theta) = \prod_{i=1}^{n}(1 - \theta)^{y_i} \cdot \theta^{1-y_i} = (1 - \theta)^{\sum_{i=1}^{n} y_i} \cdot \theta^{n - \sum_{i=1}^{n} y_i}. \tag{4.41}$$

*Consider the statistic $T(Y) = \sum_{i=1}^{n} Y_i$. Using the same steps as above, one can show that this statistic is indeed sufficient. This makes intuitive sense: To estimate $\theta$, we need only to know the* *number of times that $Y_i = 1$. Indeed, this statistic leads to the following estimator, which can easily be shown to be unbiased:*

$$\hat{\theta}(y^n) = \frac{1}{n}\sum_{i=1}^{n} y_i = \frac{1}{n}T(y). \tag{4.42}$$

*Can you extend these results to the $n$ samples from a categorical distribution?*

A sufficient statistic can be trivial. For example, the entire data sequence $Y$ is always a sufficient statistic. This motivates the definition of a **minimal** sufficient statistic:

**Definition 4.5:.** *A sufficient statistic is* **minimal** *if it is a function of any other sufficient statistic.*

In the Gaussian mean example, the entire data set $Y$ cannot be minimal, because it is not a function of the sample mean. The sample mean, however, turns out to be a minimal sufficient statistic.

Manually verifying that a statistic is sufficient is a challenging task. The following theorem provides a quick method for verifying sufficiency.

**Theorem 4.1: Neyman-Fisher Factorization.** *For a statistic $T(Y)$, if we can factor the likelihood function $p(y;\theta)$ as*

$$p(y;\theta) = g(T(y),\theta)h(y), \tag{4.43}$$

*for arbitrary functions $g(\cdot,\cdot)$ and $h(\cdot)$, then $T(Y)$ is sufficient for $\theta$.*

The proof is somewhat involved, so we omit it. The intuition here is that $T(Y)$ is sufficient if we can factor out any explicit dependence of the likelihood function on $y$. When the likelihood function factors as specified, any coupling between $\theta$ and the observation $Y$ is entirely through $T(Y)$) via $g(T(Y),\theta)$.

**Example 4.9: Gaussian mean.** *Once again consider the estimation of the mean of a Gaussian from $n$ i.i.d. samples and the statistic $T(Y) = \sum_{i=1}^{n} Y_i$. We see immediately that we can factorize the likelihood function as required:*

$$\begin{aligned} p(y|\theta) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left(n\theta^2 - 2\theta\sum_{i=1}^{n} y_i\right)\right) \times \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2\right) &(4.44)\\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left(n\theta^2 - 2\theta T(y)\right)\right) \times \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2\right) &(4.45)\\ &= g(T(y,\theta))h(y). &(4.46) \end{aligned}$$

*Thus the sum of the samples is sufficient for the mean.*

We can make a similar observation with the variance.

**Example 4.10: Gaussian variance.** *Suppose that the mean is known to be zero, but the variance $\sigma^2$ is unknown. Consider the statistic $T(Y) = \sum_{i=1}^{n} Y_i^2$. Is this a sufficient statistic? Let's look at* *the likelihood function.*

$$p(y|\sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2\right) \tag{4.47}$$

$$= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}T(y)\right) \tag{4.48}$$

$$= g(T(y), \sigma^2) \times 1. \tag{4.49}$$

*The sum of the squares of the samples is sufficient for the variance.*

We care about identifying sufficient statistics because they are they key to finding the MVUE. A key result in non-random estimation is that, under mild conditions, an unbiased estimator derived from a sufficient statistic is in fact the MVUE. To get at this result, we have to lay a bit of groundwork. For starters, we need the notion of a **complete sufficient statistic**.

**Definition 4.6:.** *We say that a sufficient statistic $T(Y)$ is* **complete** *if there is only one unbiased estimator that is a function of $T(Y)$. Equivalently, $T(Y)$ is complete if the only function $g(\cdot)$ that satisfies*

$$\int g(T(y))p(T;\theta)dy = 0, \tag{4.50}$$

*for all $\theta$ is the all-zero function $g(\cdot) = 0$, and where $p(T;\theta)$ is the distribution for the sufficient statistic $T(y)$.*

Finding and verifying a complete sufficient statistic can be tricky or tedious. Fortunately, for a large class of important distributions, they have already been found.

**Example 4.11: Exponential Families.** *Let's consider the* **exponential family** *of likelihood functions, or the distributions that take on the following form:*

$$p(y|\theta) = C(\theta)\exp\left(\sum_{l=1}^{n} Q_l(\theta)T_l(y)\right) h(y), \tag{4.51}$$

*where $\theta = (\theta_1, \ldots, \theta_n)$ is the vector of parameters, $Q_l(\theta)$ is a set of $n$ known functions on the parameters, and $T_l(y)$ is a statistic for each of the parameter functions $Q_l(\theta)$. The exponential family includes the Gaussian distribution, for $\theta = (\mu, \sigma^2)$; $Q_1(\theta) = \mu/\sigma^2$ and $Q_2(\theta) = \sigma^2$; $T_1(y) = \sum_i y_i$ and $T_2(y) = \sum_i y_i^2$. It's easy to verify that it contains many other familiar distributions, including the Poisson, Laplace, binomial, and Bernoulli distributions.*

*For exponential families, it is possible to prove that the statistics $T_l(y)$ are both sufficient and complete for the parameters $Q_l(\theta)$. In the case of the Gaussian distribution, this means that the statistics $T_1$ and $T_2$ are not only sufficient for the mean and the variance, but they are also complete. In the next theorem, we will see that this means we can find the MVUE from these statistics.*

When we have a complete sufficient statistic, it is straightforward to come up with the MVUE. This fact follows from the following theorem, called the **Rao-Blackwell theorem**.

**Theorem 4.2:.** *Let $\hat{\theta}_0(y)$ be any unbiased estimator of $\theta$, and let $T(y)$ be a sufficient statistic for $\theta$. Then, define a new estimator*

$$\hat{\theta}(y) = E[\hat{\theta}_0(y)|T(y)], \tag{4.52}$$

*or the conditional expectation of the estimator given the sufficient statistic $T(y)$. Then, the following two things are true: (1) $\hat{\theta}(y)$ is an unbiased estimator, and (2) $\mathrm{var}(\hat{\theta}) \leq \mathrm{var}(\hat{\theta}_0)$. Furthermore, if $T(y)$ is a complete sufficient statistic, then $\hat{\theta}(y)$ is the MVUE.*

First of all, the Rao-Blackwell theorem gives a recipe for improving an unbiased estimator. Take any estimator, even one with very high variance, so long as it is unbiased. Then, if we compute its expectation, conditioned on the sufficient statistic $T(y)$, we get an estimator with lower variance. Sometimes this process is (rather inelegantly) called the *Rao-Blackwellization* of an estimator. Furthermore, if $T(y)$ is a complete sufficient statistic, the resulting estimator is the MVUE. No unbiased estimator can have lower variance than the one that results from the complete sufficient statistic.

Second, the Rao-Blackwell theorem gives an explicit recipe for finding the MVUE if we have identified a complete sufficient statistic. This recipe doesn't even involve performing the Rao-Blackwellization step explicitly. Instead, if we have a complete sufficient statistic $T(y)$, find the unique function $g(\cdot)$ such that the result is an unbiased estimator:

$$\hat{\theta}(y) = g(T(y)). \tag{4.53}$$

This estimator is the unique unbiased estimator that's a function of $T(y)$, so by definition it is the one that would result by computing the Rao-Blackwell estimator. It is therefore the MVUE.

**Example 4.12: Gaussian mean and variance.** *Again consider the estimation of the mean and variance of a Gaussian distribution from $n$ i.i.d. samples. The likelihood function is*

$$p(y|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left( -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu) \right). \tag{4.54}$$

*Using the Rao-Blackwell theorem, we can find the MVUE. Recall from the above example that the Gaussian distribution is an exponential family, and if we consider the parameters $Q_1(\theta) = \mu/\sigma^2$ and $Q_2(\theta) = \sigma^2$, we have the complete sufficient statistics*

$$T_1(Y) = \sum_{i=1}^{n} Y_i \tag{4.55}$$

$$T_2(Y) = \sum_{i=1}^{n} Y_i^2. \tag{4.56}$$

*Because these statistics are complete, any unbiased estimator based on them must be the MVUE. To be very clear, these are unbiased estimators of $\mu/\sigma^2$ and $\sigma^2$. However, it is easy to convince yourself that if we can simultaneously estimate $\mu/\sigma^2$ and $\sigma^2$ with minimum variance from the same sufficient statistics. Therefore, to estimate the mean and the variance, we choose the unbiased estimators*

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} Y_i \tag{4.57}$$

$$\hat{\sigma^2} = \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \mu)^2. \tag{4.58}$$

*Because they are functions of complete sufficient statistics, we know that these estimators are MVUE.*

**Example 4.13: Bernoulli Distribution.** *Suppose we observe $n$ coin flips, denoted $Y_i$, and we want to estimate the unknown Bernoulli parameter $\theta$. The likelihood function in this case is*

$$p(y|\theta) = \theta^{\sum_{i=1}^n y_i}(1-\theta)^{n-\sum_{i=1}^n y_i} \tag{4.59}$$

*We want the MVUE of $\theta$. Fortunately, this is an exponential form, with unique (complete) sufficient statistic $T(Y) = \sum_{i=1}^n Y_i$. By the Rao-Blackwell theorem, an unbiased estimator that's a function of $T(y)$ is the MVUE. To see what the estimator should look like, let's compute the expectation of $T(y)$:*

$$E[T(Y)] = \sum_{i=1}^n E[Y_i] = n\theta. \tag{4.60}$$

*To get an* unbiased *estimator, we simply normalize by $n$, and we end up with the MVUE estimator:*

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^n Y_i. \tag{4.61}$$

*This estimator is simply the sample mean, which is exactly what one would expect.*

In non-Gaussian (or non-exponential) estimation problems, we will have to work a bit harder to find complete sufficient statistics and check to see whether or not we can derive the MVUE. One important non-exponential distribution we have already seen is the uniform distribution.

**Example 4.14: Uniform Distribution.** *Suppose we observe $n$ draws from a uniform distribution over the interval $[0,\theta]$, where the upper limit is unknown. That is, $Y_i \sim \mathcal{U}(0,\theta)$, with $\theta$ the parameter we want to estimate. The trick to finding a sufficient statistic is to find the right expression for the likelihood function. Let's write*

$$p(y|\theta) = \begin{cases} \frac{1}{\theta}, & \textit{for } 0 \le y \le \theta \\ 0, & \textit{otherwise} \end{cases} = \frac{1}{\theta}\mathbb{1}_{[0,\theta]}(y), \tag{4.62}$$

*where $\mathbb{1}_A(x)$ is the* **indicator function**, *which is equal to 1 when $x \in A$, and zero otherwise. Here, the indicator function just tells us whether $y$ is between 0 and $\theta$. Then, the likelihood function for $Y^n$ is*

$$\begin{aligned} p(y^n|\theta) &= \frac{1}{\theta}\prod_{i=1}^n \mathbb{1}_{[0,\theta]}(y_i) && (4.63)\\ &= \frac{1}{\theta}\mathbb{1}_{[-\infty,\theta]}(\max_i y_i)\mathbb{1}_{[0,\infty]}(\min_i y_i). && (4.64) \end{aligned}$$

*Now, the indicator functions check whether each $Y_i$ is smaller than $\theta$ and whether each $Y_i$ is greater than zero. By the Neyman-Fisher factorization theorem, we can identify*

$$T(y^n) = \max_i y_i \tag{4.65}$$

*as a sufficient statistic for $\theta$. This makes intuitive sense: $\theta$ must be at least as great as the maximum value of any $Y_i$ that we observe. We might be tempted to use $T(Y^n)$ as our estimator, but we need to check that it is unbiased. It is straightforward to show*[1] *that the distribution of $T(Y^n) = \max_i Y_i$*

[1] In fact, this is a good exercise to try out yourself. Learning to compute the distribution of the maximum of $n$

*is*

$$p(t|\theta) = \begin{cases} \frac{n}{\theta^n} t^{n-1}, & \text{for } 0 \leq t \leq \theta \\ 0 & \text{otherwise} \end{cases}. \tag{4.66}$$

*Then, we calculate the expected value of the statistic:*

$$E[T(Y^n)] = \int_0^\theta t \frac{n}{\theta^n} t^{n-1} dt = \frac{n}{n+1}\theta. \tag{4.67}$$

*So, this statistic* underestimates *$\theta$ on the average, and an unbiased estimator is*

$$\hat{\theta}(y^n) = \frac{n+1}{n} T(y^n) = \frac{n+1}{n} \max_i y_i. \tag{4.68}$$

*Since this estimator is a function of a sufficient statistic, it is a candidate for an MVUE. What remains is to show that $T(y^n)$ is* complete. *As seen above, this is equivalent to showing that $\hat{\theta}$ is the* only *unbiased estimator that is a function of $T(y^n)$. Proving this is a little technical, but the upshot is that this is indeed the MVUE.*

However, the MVUE may not exist.

**Example 4.15: Exponential distribution.** *Let $Y \sim \text{Exp}(\lambda)$, or $p(y|\lambda) = \lambda \exp(-\lambda y)$. This distribution is a member of the exponential family. The likelihood of $n$ i.i.d. samples is*

$$p(y^n|\lambda) = \lambda^n \exp(-\lambda \sum_i y_i). \tag{4.69}$$

*By the Neyman-Fisher factorization theorem, it's clear that $T(y^n) = \sum_i y_i$ is a sufficient statistic for $\lambda$. Furthermore, because this is a member of the exponential family, this sufficient statistic is complete. So, if we can find an unbiased estimator from $T(y^n)$, we know that it is the MVUE. To find an unbiased estimator, let's compute the expectation of the statistic:*

$$E[T(Y^n)] = \sum_{i=1}^n E[Y_i] = \frac{n}{\lambda}. \tag{4.70}$$

*The expectation of each $Y_i$ is the* reciprocal *of $\lambda$, so the sample mean has expectation $1/\lambda$ instead of $\lambda$! To convert this into an unbiased estimator, we might be temped therefore to take the reciprocal of the sample mean as our estimator, i.e. $\hat{\theta} = n/T(Y^n)$. However, this doesn't work. The expectation of the reciprocal is* not *the reciprocal of the expectation, and we do not get an unbiased estimator.*

*It turns out that there does not exist* any *unbiased estimator of $\lambda$. Therefore there can be no MVUE. Note that this is true even though we are dealing with a member of the exponential family, which we supposed above would be easiest to work with.*

### 4.1.2 Cramer-Rao Bound

If we are lucky enough to find the MVUE—if, for example, we are dealing with an exponential family whose sufficient statistics admit an unbiased estimator—we know exactly how well we can estimate the parameter $\theta$ from the observation $Y$. All we have to do is compute the variance of

i.i.d. samples is a first step in studying **order statistics**, a topic in probability in which one studies the distributions

$\hat{\theta}_{MVUE}$, and we know for certain that that's the best we can do. In more complicated scenarios, though, the MVUE is hard to compute. One surprising case of this is the exponential distribution with rate parameter $\lambda$: we can easily identify the complete sufficient statistic $\sum_{i=1}^n Y_i$ for the rate parameter $\lambda$, but it's tough to find a function of that sufficient statistic that results in an unbiased estimator.

Fortunately, there exists another, more general, way of bounding the variance of *any* unbiased estimator, even if we cannot construct one! This result, called the **Cramer-Rao bound**, depends on the *curvature* of the log-likelihood function.

**Theorem 4.3: Cramer-Rao Lower Bound.** *Suppose the regularity condition $E[\partial/\partial\theta \ln p(y;\theta)] = 0$. Let $\hat{\theta}(y)$ be any unbiased estimator of $\theta \in \mathbb{R}$ from the observation $Y$. Define the* **Fisher information** *of $\theta$ as*

$$I(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\ln p(y;\theta)\right]. \tag{4.71}$$

*Then,* $\operatorname{var}\{\hat{\theta}(y)\} \geq 1/I(\theta)$.

The intuition here is that the more that the likelihood function changes with $\theta$, the easier it is to estimate $\theta$ from the observation $Y$. Consider an extreme case: if $p(y;\theta)$ doesn't depend on $\theta$ at all, there is no way we're going to be able to estimate the parameter, because $Y$ is completely independent of $\theta$. The Cramer-Rao bound makes this intuition precise: the Fisher information $I(\theta)$ is zero, and the estimator variance is infinite. On the other hand, if the likelihood function depends heavily on $\theta$, the Fisher information is high, and the Cramer-Rao bound is small. Note that the Cramer-Rao bound is only a bound! There may not be any estimator that achieves it, but every unbiased estimator has variance at least as big as the reciprocal of the Fisher information.

**Definition 4.7:.** *If an unbiased estimator has* $\operatorname{var}\{\hat{\theta}(y)\} = 1/I(\theta)$, *we say that the estimator is* **efficient**.

Let's look again at Gaussian examples.

**Example 4.16: Gaussian mean.** *With $n$ i.i.d. samples drawn from a Gaussian with unknown mean and known variance, the likelihood function is*

$$p(y;\mu) = \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i-\mu)^2\right). \tag{4.72}$$

*The negative log-likelihood function is*

$$-\log(p(y|\mu)) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^n (y_i-\mu)^2. \tag{4.73}$$

*Taking the second derivative yields*

$$-\frac{\partial^2}{\partial\theta^2}\log p(y|\mu) = n/\sigma^2. \tag{4.74}$$

*Therefore, the expectation is superfluous, and the Fisher information is $I(\mu) = n/\sigma^2$. Therefore, for any unbiased estimator of $\mu$ we have*

$$\operatorname{var}\hat{\mu}(y) \geq \sigma^2/n. \tag{4.75}$$

of samples ordered from least to greatest.

*Of course, we already knew this, because this is exactly the variance of the MVUE.*

We can also derive the Cramer-Rao lower bound for estimating the rate of an exponential distribution, even though we couldn't derive the MVUE.

**Example 4.17: Exponential distribution.** *Suppose we obtain $n$ i.i.d. samples from an exponential distribution with rate parameter $\lambda$. The likelihood function is*

$$p(y|\lambda) = \lambda^n \exp\left(-\lambda \sum_{i=1}^{n} y_i\right), \tag{4.76}$$

*so the negative log-likelihood is*

$$-\log(p(y|\lambda)) = -n\log(\lambda) + \lambda \sum_{i=1}^{n} y_i. \tag{4.77}$$

*The first derivative is*

$$-\frac{\partial}{\partial\lambda}\log(p(y|\lambda)) = -\frac{n}{\lambda} + \sum_{i=1}^{n} y_i, \tag{4.78}$$

*so Fisher information is*

$$-\frac{\partial^2}{\partial^2\lambda}\log(p(y|\lambda)) = \frac{n}{\lambda^2}. \tag{4.79}$$

*In this case the CRLB depends on the value of $\lambda$. The smaller the value of $\lambda$, the easier it is to estimate the rate, at least in principle! It's hard to say exactly whether or not this is true, because we don't have an unbiased estimator. As we'll see later, however, the* maximum likelihood *estimator*

$$\hat{\lambda}(y) = \frac{n}{\sum_{i=1}^{n} y_i}, \tag{4.80}$$

*"nearly" approaches the CRLB, and the gap to the bound decays to zero as $n \to \infty$.*

### 4.1.3 Vector Cramer-Rao Bound

So far we have considered the case in which the parameter $\theta$ is a scalar. We also can let the parameter be a vector, i.e. $\theta = (\theta_1, \ldots, \theta_n)$. We want to derive a *vector* Cramer-Rao bound. Instead of bounding the variance of the estimate, we want to bound its covariance matrix. In order to do this, we need to define a special family of matrices.

**Definition 4.8:.** *Consider a symmetric matrix $\Sigma$. We say that $\Sigma$ is* **positive semi-definite** *if $\mathbf{x}^T\Sigma\mathbf{x} \geq 0$ for any vector $\mathbf{x}$. Equivalently, $\Sigma$ is positive semi-definite if its eigenvalues are non-negative.*

Every covariance matrix is positive semi-definite; this is true by the definition of the covariance matrix. We use the definition of the positive semi-definite matrix to indicate when a matrix is "bigger" than another. We formalize this notion by the following definition.

**Definition 4.9:.** *For two positive semi-definite matrices $\mathbf{A}$ and $\mathbf{B}$, we say that $\mathbf{A} \geq \mathbf{B}$ provided the matrix $\mathbf{A} - \mathbf{B}$ is positive semi-definite.*

What does this mean? If $\mathbf{A}$ and $\mathbf{B}$ are diagonal, then $\mathbf{A} \geq \mathbf{B}$ exactly when all of the elements of $\mathbf{A}$ are greater than all of the elements of $\mathbf{B}$. When $\mathbf{A}$ and $\mathbf{B}$ are *not* diagonal, the inequality is more complicated. But intuitively, if two covariance matrices $\mathbf{A}$ and $\mathbf{B}$ satisfy $\mathbf{A} \geq \mathbf{B}$, then $\mathbf{A}$ expresses more total variance than $\mathbf{B}$.

With these definition in place, we can define the vector Cramer-Rao bound.

**Theorem 4.4: Vector Cramer-Rao Lower Bound.** *Again suppose the regularity condition $E[\partial/\partial\theta \log p(y;\theta)] = 0$, where here we mean that the gradient vector is zero. Then, define the* **Fisher information matrix** *of the vector parameter $\theta$, where*

$$[I(\theta)]_{ij} = -E\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log p(y;\theta)\right]. \tag{4.81}$$

*Then, the covariance of the estimator satisfies*

$$\Sigma_{\hat{\theta}} \geq I^{-1}(\theta), \tag{4.82}$$

*where $I^{-1}(\theta)$ is the inverse of the Fisher information matrix.*

Let's work this out for the case of Gaussian mean and variance.

**Example 4.18: Gaussian mean and variance.** *Suppose that the mean and variance of the Gaussian distribution are unknown. Then, $\theta = (\mu, \sigma^2)$, and the negative log-likelihood function is*

$$-\log p(y|\theta) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2. \tag{4.83}$$

*It's straightforward to compute the Fisher information matrix:*

$$I(\theta) = \begin{bmatrix} n/\sigma^2 & 0 \\ 0 & n/(2\sigma^4) \end{bmatrix}. \tag{4.84}$$

*The Fisher information matrix is diagonal, which means that the CLRB predicts estimation error that is uncorrelated. Therefore we can bound the variance of each parameter estimate individually:*

$$\mathrm{var}(\hat{\mu}) = \frac{\sigma^2}{n} \tag{4.85}$$

$$\mathrm{var}(\hat{\sigma}^2) = \frac{2\sigma^4}{n}. \tag{4.86}$$

*The first variance bound is indeed achieved by the MVUB. Is the second?*

Let's look at an example in which we'll see correlations in the CRLB.

**Example 4.19: Line fitting.** *Consider fitting noisy data to a line:*

$$Y_i = A + Bi + Z_i, i = 1, \ldots, n, \tag{4.87}$$

*where $A$ and $B$ are unknown constants, and $Z_i \sim \mathcal{N}(0, \sigma^2)$ for known variance $\sigma^2$. We want to estimate the slope and intercept defined by $\theta = (A, B)$. This is equivalent to* **linear regression**, *where we fit data points to a linear model assuming that they are corrupted by white Gaussian noise. The likelihood function is*

$$p(y|\theta) = \frac{1}{(2\pi\sigma^2)^{(n/2)}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - A - Bi)^2\right). \tag{4.88}$$

*From this, we can compute the Fisher information matrix:*

$$I(\theta) = \frac{1}{\sigma^2} \begin{bmatrix} n & \sum_{i=1}^n i \\ \sum_{i=1}^n i & \sum_{i=1}^n i^2 \end{bmatrix} \tag{4.89}$$

$$= \frac{1}{\sigma^2} \begin{bmatrix} n & \frac{n(n-1)}{2} \\ \frac{n(n-1)}{2} & \frac{n(n-1(2n-1))}{6} \end{bmatrix} \tag{4.90}$$

*This is an easy matrix to invert:*

$$I(\theta) = \sigma^2 \begin{bmatrix} \frac{2(2n-1)}{n(n+1)} & -\frac{6}{n(n+1)} \\ -\frac{6}{n(n+1)} & \frac{12}{n(n^2-1)}. \end{bmatrix} \tag{4.91}$$

*Therefore, we get the bounds*

$$\mathrm{var}(\hat{A}) \geq \frac{2(2n-1)}{n(n+1)} \tag{4.92}$$

$$\mathrm{var}(\hat{B}) \geq \frac{12}{n(n^2-1)}. \tag{4.93}$$

*In addition to these individual variances, we see that the estimates are negatively correlated. It turns out to be easier to estimate $B$ than $A$.*

Finally, we can also discuss general approaches to the CRLB.

**Example 4.20: Signals in white Gaussian noise.** *Suppose we have a family of signals parameterized by a scalar $\theta$, i.e. the signal $\mathbf{s}(\theta)$ is an $n$-dimensional vector (or a signal with $n$ samples) where the unknown parameter $\theta$ specifies the signal from the family. In this case, the likelihood function is*

$$p(y|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left( -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - s_i(\theta))^2 \right). \tag{4.94}$$

*Differentiating the log-likelihood function, we get*

$$\frac{\partial \log p(y|\theta)}{\partial \theta} = \frac{1}{\sigma^2} \sum_{i=1}^n (y_i - s_i(\theta)) \frac{\partial s_i(\theta)}{\partial \theta}. \tag{4.95}$$

*Taking another derivative gives us*

$$\frac{\partial^2 \log p(y|\theta)}{\partial \theta^2} = \frac{1}{\sigma^2} \sum_{i=1}^n \left[ (y_i - s_i(\theta)) \frac{\partial^2 s_i(\theta)}{\partial \theta^2} - \left( \frac{\partial s_i(\theta)}{\partial \theta} \right)^2 \right]. \tag{4.96}$$

*Therefore, the Fisher information is*

$$I(\theta) = \frac{1}{\sigma^2} \sum_{i=1}^n \left( \frac{\partial s_i(\theta)}{\partial \theta} \right)^2, \tag{4.97}$$

*and the CRLB is*

$$\mathrm{var}(\hat{\theta}) \geq \frac{\sigma^2}{\sum_{i=1}^n \left( \frac{\partial s_i(\theta)}{\partial \theta} \right)^2}. \tag{4.98}$$

### 4.1.4 Maximum-likelihood Estimation

A major challenge so far has been that we cannot always find the MVUE. In those cases, we can compute the CRLB to bound the variance of an unbiased estimator, but what estimator should we use? Are we stuck guessing at an estimator and hoping for the best?

Sort of. define the **maximum-likelihood (ML) estimator**:

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(y|\theta). \tag{4.99}$$

This estimator makes heuristic sense: The value of $\theta$ that maximizes the likelihood is the one that is perhaps most consistent with the data. In absence of an unbiased estimator, it seems a worthwhile choice. But we cannot say that it satisfies any optimality conditions in terms of bias or variance. Indeed, the ML estimator is not, in general, the MVUE or even an unbiased estimator!

To find the ML estimate, we'll take the derivative of the log-likelihood function and set it equal to zero:

$$\frac{\partial}{\partial \theta} \log p(y|\theta) = 0, \tag{4.100}$$

which is often called the **likelihood equation**. As long as the likelihood function is smooth and concave in $\theta$, the solution to the likelihood function is the ML estimator. Otherwise, satisfying the likelihood equation not a sufficient condition for $\hat{\theta}_{ML}$. We'll spend most of our time looking at examples for which the likelihood equation is sufficient.

**Example 4.21: Gaussian estimation.** *As usual, we'll look at the estimation of the mean and variance from Gaussian observations:*

$$p(y|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left( -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2 \right). \tag{4.101}$$

*It's straightforward to show that the ML estimates of the mean and variance are*

$$\hat{\mu}_{ML} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{4.102}$$

$$\hat{\sigma^2}_{ML} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{\mu}_{ML})^2. \tag{4.103}$$

*The ML estimate of the mean is unbiased—and indeed it is the MVUE—but the estimate of the variance is* not *unbiased!*

What is going on in this case? The ML estimator of the variance is indeed biased; on the other hand, its variance is equal to that predicted by the CRLB, which is lower than the variance of the MVUE of the variance. This example exposes an interesting fact about the ML estimate, which we report in the following theorem.

**Theorem 4.5:.** *If an unbiased estimator is efficient—i.e., it achieves the Cramer-Rao bound—then it is the maximum-likelihood estimator.*

As usual, we skip the proof. However, since the CRLB is connected to taking derivatives of the likelihood function, it's not too hard to convince yourself that the ML estimator should be connected to the CRLB. In the case of the Gaussian mean and variance, the CRLB can be achieved for the estimate of the mean, but not the variance. Therefore the ML estimate of the mean is the same as the MVUE, but the ML estimate of the variance is not.

**Example 4.22: Exponential distribution.** *Suppose we obtain $n$ i.i.d. samples from an exponential distribution with rate parameter $\lambda$. The likelihood function is*

$$p(y|\lambda) = \lambda^n \exp\left(-\lambda \sum_{i=1}^{n} y_i\right), \tag{4.104}$$

*so the log-likelihood is*

$$\log(p(y|\lambda)) = -n\log(\lambda) + \lambda \sum_{i=1}^{n} y_i. \tag{4.105}$$

*The likelihood function is*

$$\frac{\partial}{\partial\lambda}\log(p(y|\lambda)) = \frac{n}{\lambda} - \sum_{i=1}^{n} y_i = 0, \tag{4.106}$$

*so the ML estimate is*

$$\hat{\lambda}_{ML} = \frac{n}{\sum_{i=1}^{n} y_i}, \tag{4.107}$$

*or the reciprocal of the sample mean. This estimator, too, is biased. Recall that no estimator achieves the CRLB for this case, because we could not construct an unbiased estimator. This estimator is "almost" unbiased, in a way that we'll make precise shortly.*

**Definition 4.10:.** *Let $\hat{\theta}_n$ be an estimator based on $n$ samples of a random variable. The sequence of estimators $\{\hat{\theta}_n\}$ is said to be* **consistent** *if*

$$\hat{\theta}_n \to \theta, \tag{4.108}$$

*where the convergence is* almost sure *convergence, i.e. convergence with probability one.*

In other words, a consistent estimator is one that, given enough data, returns a value close to $\theta$. We'd hope that we can always find a consistent estimator. After all, with more and more data, we ought to be able to find a better and better estimate, to the point that the uncertainty decays to zero in the asymptote. The following theorem states that, via the ML estimator, we can!

**Theorem 4.6:.** *Let $\hat{\theta}_n$ be the ML estimator using $n$ i.i.d. samples of $Y$ from the likelihood function $p(y|\theta)$. The sequence of estimators $\{\hat{\theta}_n\}$ is consistent.*

The ML estimator may not be unbiased or efficient, but in the limit of sufficiently many samples, it converges almost surely on $\theta$. The ML estimator is therefore *asymptotically* unbiased!

It turns out that the ML estimator is asymptotically efficient, too.

**Theorem 4.7:.** *Let $\hat{\theta}_n$ be the ML estimator using $n$ i.i.d. samples of $Y$ from the likelihood function $p(y|\theta)$. Let the likelihood function be such that the Fisher information $I(\theta)$ exists and is invertible. Then, if $\theta_0$ is the true value of the parameter, then the asymptotic distribution of $\hat{\theta}_n$ satisfies*

$$\lim_{n\to\infty} \sqrt{n}(\hat{\theta}_n - \theta_0) - \sim \mathcal{N}(0, I^{-1}(\theta_0)). \tag{4.109}$$

In other words, in the limit of many samples, the ML estimate approaches a Gaussian distribution, with the mean at the true value $\theta_0$, and the covariance approximately equal to $1/nI^{-1}(\theta_0)$. This is exactly the variance predicted by the CRLB.

Therefore, if we have enough samples, the ML estimate is a rather good option!

Another important case of maximum-likelihood estimation is the recovery of a signal from noisy linear measurements.

**Example 4.23: Linear Measurements/Noisy Deconvolution.** *Suppose that we want to estimate a signal* $\mathbf{x} \in \mathbb{R}^d$ *that we observe through noisy linear measurements:*

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}, \tag{4.110}$$

*where* $\mathbf{H} \in \mathbb{R}^{m \times d}$ *is a matrix and* $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ *is Gaussian noise. The matrix* $\mathbf{H}$ *indicates that the signal to estimate is passed through a linear system, which we suppose is known. For example, if the system is linear time-invariant with impulse response* $h = (h_1, h_2, \cdots, h_d)$*, then the matrix*

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_d \\ h_2 & h_3 & h_4 & \cdots & h_1 \\ & & \vdots & & \\ h_d & h_1 & h_2 & \cdots & h_{d-1} \end{bmatrix} \tag{4.111}$$

*represents the convolution of* $\mathbf{x}$ *with the impulse response. This might represent channel that a wireless signal goes through in traveling over the air or the blurring of an image. In this case, we call the estimation of* $\mathbf{x}$ *from* $\mathbf{y}$ **noisy de-convolution** *problem, since the task is to reverse the convolution from measurements corrupted by noise. Maximum-likelihood estimation is a simple approach to de-convolution, and we'll see more sophisticate approaches when we look at Bayesian estimation.*

*The likelihood function is Gaussian:* $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{H}\mathbf{x}, \sigma^2\mathbf{I})$*. The log-likelihood is*

$$\log(p(\mathbf{y}|\mathbf{x})) = -\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{H}\mathbf{x})^T(\mathbf{y} - \mathbf{H}\mathbf{x}) + c = \frac{1}{\sigma^2}\mathbf{x}^T\mathbf{H}^T\mathbf{y} - \frac{1}{2\sigma^2}\mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x} + \frac{1}{2\sigma^2}\mathbf{y}^T\mathbf{y}. \tag{4.112}$$

*Taking the gradient with respect to* $\mathbf{x}$*, we can find the conditions for the ML estimator:*

$$\nabla_{\mathbf{x}} \log(p(\mathbf{y}|\mathbf{x})) = \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{y} - \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H}\mathbf{x}. \tag{4.113}$$

*Setting this equal to zero, we get the conditions*

$$\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{y}. \tag{4.114}$$

*To solve, we need to make some assumptions about* $\mathbf{H}$*. If* $\mathbf{H}$ *is square and invertible, then so too is* $\mathbf{H}^T\mathbf{H}$*, and we can solve by multiplying both sides by* $(\mathbf{H}^T\mathbf{H})^{-1}$*, which gives the very simple estimator*

$$\hat{\mathbf{x}} = \mathbf{H}^{-1}\mathbf{y}. \tag{4.115}$$

*If* $m > n$*, then* $\mathbf{H}$ *is a* tall *matrix, and it cannot be invertible. If we suppose that the* columns *of* $\mathbf{H}$ *are linearly independent, then* $(\mathbf{H}^T\mathbf{H})^{-1}$ *is still invertible, and the ML estimator is*

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}. \tag{4.116}$$

*If* $m < bn$*, then* $\mathbf{H}$ *is a* fat *matrix and again cannot be inverted. If the rows are independent, we get a similar solution:*

$$\hat{\mathbf{x}} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}\mathbf{y}. \tag{4.117}$$

*It turns out we can unify these estimators under a single definition. Regardless of whether* $\mathbf{H}$ *is square, or whether its rows and columns are independent, we can write the ML estimator as*

$$\hat{\mathbf{x}} = \mathbf{H}^{+}\mathbf{y}, \tag{4.118}$$

*where* $\mathbf{H}^{+}$ *is the* **pseudo-inverse** *of* $\mathbf{H}$.

*We are usually interested in cases in which* $\mathbf{H}$ *is not invertible, in which case the ML estimator cannot provide an accurate estimate of* $\mathbf{x}$*. When we discuss Bayesian estimation, we will see how we can use a prior distribution to encode information about the structure of the signal that will substantially improve estimator performance.*

## 4.2 Introduction to Machine Learning

Here we'll apply what we have learned about hypothesis testing and estimation to a subject that puts them together: **machine learning**. Machine learning is a field that is notortiously difficult to pin down, and which has close connections to statistics, signal processing, and computer science. Lacking a single, comprehensive definition, we'll try out the following: Machine learning is the learning of an inferential model when the *underlying probability distribution of the data is unknown* and must be learned from *training data*. Up until now, we've supposed that we know the likelihood function $p(y|h)$ or $p(y|\theta)$. In machine learning, we don't know the data distribution, but instead have access to training data to help us build our inferential model. The scope of machine learning problems is vast, and we'll focus on a few key examples.

Machine learning problems are often split into two categories: **supervised learning** and **unsupervised learning**. In supervised learning, one has an input signal $X$ and and output signal $Y$, and the objective is to learn a model that maps the inputs to outputs from so-called *labeled* training samples $(X, Y)$ of both inputs and outputs. We call this "supervised" learning because of the availability of the label $Y$, which "supervises" the resulting model to make sure it agrees with the training labels. Examples of supervised learning problems are:

- **Classification.** Let $X$ be a data sample (such as the image of a face), and let $Y$ be its **label**, or the class to which $X$ belongs (such as the identity of the person photographed). The objective is to learn a **classifier** that maps an input $X$ to labels $Y$ from so-called labeled pairs $(X, Y)$. In this case, the inference problem is simply multiple-hypothesis testing, and the learning task is to find a data model that allows us to carry out hypothesis testing as we studied before. Important classification models include **support vector machines**, **deep neural networks**, and **logistic regression**. We will study logistic regression in this chapter.
- **Regression and Prediction.** Let $X$ be a data sample (such as a patient's clinical data—height, weight, age, etc.) and $Y$ be quantities that we want to predict from $X$ (such as the size and progression of a patient's tumor). The objective is to learn a mapping from $X$ to $Y$ that allows us to predict $Y$ from $X$, so in this case the objective is the estimation of $Y$ given $X$. We call this the **regression** or **prediction** problem. Important models include **linear regression**, which we will study in this chapter, and **time-series models** such as the auto-regressive moving average (ARMA), which we will not.

In unsupervised learning, one has only the signal $X$, and the objective is to learn a data model that gives a useful representation of the data. Here, there are no labels $Y$ to "supervise" the process. Examples of unsupervised learning problems are

- **Density estimation.** Given a collection of samples $X_i$, we want to estimate the probability density function $p(x)$. There are myriad techniques for this, including **kernel density estimation**.
- **Feature extraction.** Given a collection of samples $X_i$, we want to extract a **features** that give a compact, meaningful description of the data. Important feature extraction techniques include **principal components analysis** and **dictionary learning**.
- **Clustering.** Even without labels, we may be able to sort data samples $X_i$ into classes that are inherent in the data. This is the clustering problem, and it can be thought of as an unsupervised approach to learning a classifier. Important clustering techniques include **k-means clustering** and the **expectation-maximization algorithm**.

It's important to note that these boundaries are not as crisp as the above discussion might suggest. **Semi-supervised learning** is carried out in the case in which only some of the data samples have labels. Some feature extraction problems are supervised—indeed, deep learning can be thought of as a feature extraction problem where the challenge is to find features that are most useful for classification. Nevertheless, we will take for granted the schema that we've described here.

### 4.2.1 Posing a Machine Learning Problem: Empirical Risk Minimization

Now that we have an intuitive sense of what we're trying to do in machine learning, let's pose the problem formally. We suppose that we are given access to $n$ **training samples** $(X_i, Y_i)$, $1 \le i \le n$ in the case of supervised learning and $X_i$, $1 \le i \le n$ in the case of unsupervised learning. These data are drawn i.i.d. from some distribution $p(x, y)$ or $p(x)$ that is unknown to us.

The objective of learning is to choose some function $f(x)$ that maps an input sample $X$ to an output—a class label in the case of classification, or a set of features in the case of feature extraction. In the abstract, we define the quality of the function $f(x)$ that we've learned in terms of the **loss function** $l(f(x), y)$ (or $l(x)$ in the case of unsupervised learning). For example, in classification, a suitable loss function is the so-called *0-1 loss*:

$$l(f(x), y) = \begin{cases} 0, & \text{if } f(x) = y \\ 1, & \text{if } f(x) \neq y \end{cases}, \tag{4.119}$$

which returns a 0 if $f(x)$ classifies the signal correctly, and 1 if not. For regression, a suitable loss function is the *squared error loss*:

$$l(f(x), y) = \|f(x) - y\|^2, \tag{4.120}$$

which returns the squared norm of the prediction error.

The loss function is pointwise, giving us the loss we suffer for individual pairs $x$ and $y$. Ideally, we'd like to minimize the *expected* loss, averaged over the data distribution $p(x, y)$. We call this the **risk**:

$$L(f) = E_{X,Y}[l(f(x), y)]. \tag{4.121}$$

Unfortunately, we don't *know* the data distribution $p(x, y)$, or else there would be nothing to learn! Instead, we will minimize our best approximation of the expected value, which is the *empirical* average over the training samples. We call this this the **empirical risk**:

$$L_n(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i). \tag{4.122}$$

In choosing the function $f(x)$, we will follow the policy of *empirical risk minimization* (ERM):

$$f^*(x) = \arg\min_f L_n(f). \tag{4.123}$$

To find the ERM solution, we will follow the optimization procedures we discussed in Chapter 2, often appealing to gradient descent. When the data set is very large, an extremely popular approach to minimizing the empirical risk is **stochastic gradient descent** (SGD). In SGD, we take the gradient of the loss averaged over only a few samples at a time—even as few as one! SGD makes computing the gradients easier, since we only use a few data samples at a time. One can also show that it has nice convergence properties.

### 4.2.2 Evaluating a Machine Learning Solution: Generalization Error and Cross-validation

A major issue in machine learning is **overfitting**. If we fit a very complex learning model to only a few data points, we might end up with a model that performs quite well on the training set, but badly in general. In such a case, the empirical risk $L_n$ over the training set will be small, but the average risk $L$ will be high, and we will observe poor performance when we apply the model to data outside the training set. In the parlance of machine learning, we say that model performance does not **generalize** well to new data points. We quanitify this phenomenon by defining the **generalization error**:

$$G(f) = L(f) - L_n(f). \tag{4.124}$$

We can't compute the generalization error directly—doing so would require knowledge of $p(x, y)$, which would mean we could learn $f(x)$ directly!

Instead, a ubiquitious solution in machine learning is **cross-validation**. Instead of using all $n$ samples to train a model, one "leaves out" a fraction of the samples, usually around 10-20%. One trains the model using the remaining 80-90% of the model via empirical risk minimization. Then, to estimate the risk of the model, one computes the empirical risk over the held-out samples. Because these samples were not used to choose the model $f(x)$, they are "honest" random samples, and their empirical risk is a reasonably good estimate of $L(f)$. A more sophisticated variation is **k-fold cross validation**, in which the training set is partitioned randomly into $k$ sets of equal size, where $k$ is usually between 5 and 10. Then, one trains $k$ models in succession, leaving out one of the sets, training on the $k-1$ remaining sets, and evaluating the empirical error over the held-out set. The average error over all $k$ models is taken as an estimate of the true risk $L(f)$.

In some cases, we can prove theoretical bounds on the generalization error. We won't go into these methods in any detail, as they require quite a bit of technical machinery. However, the upshot is that, by arguments related to the law of large numbers and central limit theorem, in many cases the generalization error decays as $1/\sqrt{n}$.

### 4.2.3 Supervised Learning

#### 4.2.3.1 Linear Regresssion

In linear regression, we suppose we have a vector input signal $\mathbf{x} \in \mathbb{R}^d$ that maps to a vector output signal $\mathbf{y} \in \mathbb{R}^p$, where $p$ and $d$ are not necessarily the same. The regression task is to learn a *linear* function $f(x)$ that maps input to output:

$$f(\mathbf{x}) = \mathbf{B}\mathbf{x} + \mathbf{c}, \tag{4.125}$$

where $\mathbf{B} \in \mathbb{R}^{p\times d}$ is a matrix that describes the linear relationship between $\mathbf{x}$ and $\mathbf{y}$, and $\mathbf{c} \in \mathbb{R}^p$ is a constant offset. A common approach is to reparameterize the regression function by letting $\mathbf{x}$ be a $d+1$-dimensional vector, where we append a one to the end, and then let $\mathbf{B} \in \mathbb{R}^{p\times d+1}$ be the matrix that has $\mathbf{c}$ as its last column. Then we can rewrite $f(\mathbf{x}) = \mathbf{B}\mathbf{x}$.

Our objective is to choose $\mathbf{B}$ to minimize the squared error loss:

$$l(f(\mathbf{x}), \mathbf{y}) = \|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2 = (\mathbf{y} - \mathbf{B}\mathbf{x})^T(\mathbf{y} - \mathbf{B}\mathbf{x}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{B}^T\mathbf{y} + \mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x}. \tag{4.126}$$

We suppose that we have a set of $n$ labeled training pairs $(\mathbf{x}_i, \mathbf{y}_i)$. Then, the empirical risk is

$$L_n(f) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{B}\mathbf{x}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{y}_i^T\mathbf{y}_i - 2\mathbf{x}_i^T\mathbf{B}^T\mathbf{y}_i + \mathbf{x}_i^T\mathbf{B}^T\mathbf{B}\mathbf{x}_i\right). \tag{4.127}$$

If we let $\mathbf{X} = [\mathbf{x}_1\mathbf{x}_2\cdots\mathbf{x}_n] \in \mathbb{R}^{d\times n}$ be the input data matrix and $\mathbf{Y} = [\mathbf{y}_1\cdots\mathbf{y}_n]$ be the output data matrix, we can rewrite the empirical loss in the following compact form:

$$L_n(f) = \mathrm{trace}[(\mathbf{Y} - \mathbf{B}\mathbf{X})^T(\mathbf{Y} - \mathbf{B}\mathbf{X})], \tag{4.128}$$

where trace$(\cdot)$ is the matrix trace, which sums up the diagonal elements of a matrix. At this point, we can use well-known matrix-vector gradient identities to find that

$$\nabla_{\mathbf{B}} L_n(f) = \mathbf{Y}\mathbf{X}^T - \mathbf{B}\mathbf{X}\mathbf{X}^T. \tag{4.129}$$

Setting the gradient to zero, and solving, the ERM solution is

$$\mathbf{B}^* = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}, \tag{4.130}$$

in the case that $\mathbf{X}\mathbf{X}^T$ is an invertible matrix, which is true almost surely as long as $n \geq d$. Otherwise, the solution is the pseudoinverse $B^* = \mathbf{Y}(\mathbf{X}^T)^+$ of the transpose of the data matrix.

Note the close similarity of this problem to the deconvolution problem described earlier. Indeed, one can think of linear regression as a linear estimation problem in which the data follow the model:

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{z}, \tag{4.131}$$

where $\mathbf{z}$ is zero-mean noise. In particular, if $\mathbf{z} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, the maximum-likelihood estimate of $\mathbf{B}$ from $n$ data samples is identical to the linear regression solution derived above. In the previous case, we supposed we knew the linear relationship and wanted to estimate the input from the output. Here, we know the inputs and outputs, and we want to estimate the linear relationship between the inputs and outputs.

It turns out that linear regression as stated above is a good idea even if the noise is not Gaussian. An important result in statistics called the **Gauss-Markov theorem** states that the linear regression solution has the minimum variance among all *unbiased, linear* estimators, regardless of the underlying data distribution. This is a powerful result: If we restrict our attention to linear estimators, the minimum-variance solution is the one we get if we suppose that the data are Gaussian. This estimator, often called the **best linear unbiased estimator** (BLUE), is often used when the MVUE is not available.

However, in the case $n < d$ of fewer samples than data dimension, the model learned by linear regression performs quite poorly. This issue is commonplace in high-dimensional signal processing. In this case, we will encode model structure into a prior distribution, which will regularize the regression problem and allow for inference even when we the size of the training set is limited.

#### 4.2.3.2 Logistic Regression

Our objective is to learn a classifier that maps data points $\mathbf{x} \in \mathbb{R}^d$ to class labels $y \in \{1, \ldots, M\}$. The job of the classifier is to carry out multiple hypothesis testing; here $y$ denotes which hypothesis is true. We make this shift in notation to keep consistent with the rest of the section.

In logistic regression, we learn a *linear* classifier. Logistic regression is situated in the Bayesian decision framework. Recall that in Bayesian decision theory, if the objective is to minimize the probability of error, then the optimum classifier is the maximum *a priori* classifier, which chooses $\hat{y} = \arg\max_y p(y|\mathbf{x})$. To make a linear detector, we suppose that the *ratio of the log posteriors* is a linear function:

$$\log \frac{p(Y=0|\mathbf{x})}{p(Y=M-1|\mathbf{x})} = b_0 + \mathbf{b}_0^T\mathbf{x} \tag{4.132}$$

$$\log \frac{p(Y=1|\mathbf{x})}{p(Y=M-1|\mathbf{x})} = b_1 + \mathbf{b}_1^T\mathbf{x} \tag{4.133}$$

$$\vdots \tag{4.134}$$

$$\log \frac{p(Y=M-2|\mathbf{x})}{p(Y=M-1|\mathbf{x})} = b_{M-2} + \mathbf{b}_{M-2}^T\mathbf{x}. \tag{4.135}$$

These ratios are called the **log odds**, and are the Bayesian analogue to the log-likelihood ratio. To classify a point $\mathbf{x}$, one simply computes all $M-1$ log odds. If they are all less than zero, then $\hat{y} = M$ is the MAP classification of $\mathbf{x}$. Otherwise, the class with the largest log odds is the MAP classification.

Equivalently, logistic regression supposes that the posterior distribution is of the form

$$p(y|\mathbf{x}) = \frac{\exp(b_i + \mathbf{b}_i^T\mathbf{x})}{1 + \sum_{l=1}^{M-1} \exp(b_l + \mathbf{b}_l^T\mathbf{x})} \tag{4.136}$$

for $0 \leq y \leq M-2$, and

$$p(y = M-1|\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{M-1} \exp(b_l + \mathbf{b}_l^T\mathbf{x})}, \tag{4.137}$$

for $y = M-1$. In the case of $M = 2$, the posterior is the *logistic sigmoid* of the linear function $b_1 + \mathbf{b}_1^T\mathbf{x}$, which is where logistic regression gets its name. As in the case of linear regression, we will lump $b_i$ into $\mathbf{b}_i$ by concatenating a '1' to $\mathbf{x}$, giving the linear function $\mathbf{b}_i^T\mathbf{x}$.

Recall that we have seen linear classifiers and posterior distributions of this form before. When the likelihood functions are multivariate Gaussians with identical covariances, the resulting classifier is a linear hyperplane and the posterior is indeed of the form above. However, in logistic regression we do not suppose Gaussian likelihoods. In fact, we do not concern ourselves with the likelihood functions at all. Instead, we suppose that the posterior has this logistic form, which is consistent with many other possible likelihoods.

Given a set of labeled data $(\mathbf{x}_i, y_i)$, $1 \leq i \leq n$, we want to learn the parameters $b_i$ and $\mathbf{b}_i$ that define the classifier. The cost function in this case is the *negative log-likelihood* for the learned posterior:

$$l(f(\mathbf{x}), y) = -\log(p(y|\mathbf{x})). \tag{4.138}$$

That is, we treat the class label, conditioned on $\mathbf{x}$, as a categorical distribution with unknown parameters, and we wish to learn these parameters via maximum likelihood estimation. in this way, logistic regression combines Bayesian detection with maximum-likelihood estimation.

In the case of $M = 2$, the cost function is particularly easy to write out.

$$l(f(\mathbf{x}), y) = -\log(p(y|\mathbf{x})) \tag{4.139}$$

$$= \begin{cases} -\mathbf{b}^T\mathbf{x} + \log(1 + \exp(\mathbf{b}^T\mathbf{x})), & \text{for } y = 0 \\ \log(1 + \exp(\mathbf{b}^T\mathbf{x})), & \text{for } y = 1 \end{cases} \tag{4.140}$$

$$= -(y-1)\mathbf{b}^T\mathbf{x} + \log(1 + \exp(\mathbf{b}^T\mathbf{x})). \tag{4.141}$$

The empirical risk is therefore

$$L_n(f) = -\frac{1}{n}\sum_{i=1}^{n}((1-y_i)\mathbf{b}^T\mathbf{x}_i - \log(1 + \exp(\mathbf{b}^T\mathbf{x}_i))). \tag{4.142}$$

It turns out that we cannot minimize the empirical risk in closed form, as we could for linear regression. Instead, we will compute the gradient and use (stochastic) gradient descent:

$$\nabla_{\mathbf{b}} L_n(f) = -\frac{1}{n}\mathbf{x}_i\left((1-y_i) - \frac{\exp(\mathbf{b}^T\mathbf{x}_i)}{1 + \exp(\mathbf{b}^T\mathbf{x}_i)}\right). \tag{4.143}$$

The gradient descent steps have the form

$$\mathbf{b}_{t+1} = \mathbf{b}_t + \alpha_t\frac{1}{n}\mathbf{x}_i\left((1-y_i) - \frac{\exp(\mathbf{b}^T\mathbf{x}_i)}{1 + \exp(\mathbf{b}^T\mathbf{x}_i)}\right), \tag{4.144}$$

) where $\alpha_t$ is the step size. The log-likelihood function is concave, so gradient descent is guaranteed to converge on a global optimum of $L_n(f)$ so long as the step size is chosen carefully.

In addition to using $\mathbf{b}$ as a linear classifier, we can use it to interpret the dependence of the classification decision on each element of $\mathbf{x}$. If $\mathbf{b}$ is close to zero for a particular element, then it is not important in discriminating classes.

For $M > 2$, it is possible to derive a similar gradient update rule, although its form and derivation are more involved. Logistic regression is well understood and implemented in most data science languages, including scikit-learn.

Logistic regression is just one approach to finding a linear classifier. Two other very popular methods are **linear discrimniant analysis** (LDA) and the **support vector machine** (SVM). They also fit a linear classifier to data, but they use different assumptions and therefore choose the classifier according to different criteria. However, the basic spirit of the approach is the same.

### 4.2.4 Unsupervised Learning

#### 4.2.4.1 Principal Components Analysis

Principal components analysis (PCA) is the single most-commonly used technique for extracting representative features from data. Given a data point $\mathbf{x} \in \mathbb{R}^d$, we want to find a linear, $k$-dimensional representation:

$$f(\mathbf{x}) = \mathbf{V}(\mathbf{x} - \mu), \tag{4.145}$$

where $\mathbf{V} \in \mathbb{R}^{k \times p}$ is an orthonormal matrix, and $k < d$. That is, we take (a translation of) $\mathbf{x}$ and project it onto a $k$-dimensional subspace. The objective is to find the translation vector and subspace basis that minimizes the squared error of the reconstruction. Since $\mathbf{V}$ is taken to be orthonormal, $\mathbf{V}^T$ is its pseudo-inverse, so the optimum reconstruction of $\mathbf{x}$ from $f(\mathbf{x})$ is

$$\mathbf{V}^T f(\mathbf{x}) + \mu = \mathbf{V}^T \mathbf{V}(\mathbf{x} - \mu) + \mu. \tag{4.146}$$

The squared reconstruction loss function is

$$l(f(\mathbf{x})) = \left\| \mathbf{x} - \mu + \mathbf{V}^T \mathbf{V}(\mathbf{x} - \mu) \right\|^2, \tag{4.147}$$

and the empirical loss is

$$L_n(f) = \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \mu + \mathbf{V}^T \mathbf{V}(\mathbf{x}_i - \mu) \right\|^2. \tag{4.148}$$

This loss function is convex in $\mu$, and setting the gradient equal to zero yields the optimum translation vector

$$\mu^* = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i, \tag{4.149}$$

which is just the sample mean.

The loss is not convex in $\mathbf{V}$. However, one can show that it has a closed form solution. First, form the empirical covariance matrix of the data samples:

$$\mathbf{K} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \mu^*)(\mathbf{x}_i - \mu^*)^T. \tag{4.150}$$

Next, take its eigendecomposition:

$$\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{U}^T. \tag{4.151}$$

Recall that $\mathbf{K}$ is positive semi-definite, which means that the eigenvector matrix $\mathbf{U} \in \mathbb{R}^{d \times d}$ is orthonormal and the eigenvalues in $\mathbf{D}$ are nonnegative. The optimum solution is to choose $\mathbf{V}^*$ to be the $k$ eigenvectors corresponding to the largest eigenvalues of $\mathbf{K}$.

The standard interpretation of PCA is that we "center" the data by subtracting away the mean, then we project the data onto the subspace that best preserves the energy in the signal, which turns out to be the dominant eigenspace of the covariance. We can also derive PCA from maximum likelihood principles. If we suppose that the data has a Gaussian distribution with unknown mean and a covariance that is "nearly" rank $k$, we get the likelihood function

$$p(\mathbf{x}|\mu, \mathbf{V}) \propto \exp(\frac{1}{2}(\mathbf{x} - \mu)(\mathbf{V}^T \mathbf{D} \mathbf{V} + \sigma^2 \mathbf{I})^{-1} (\mathbf{x} - \mu)^T), \tag{4.152}$$

where $\mathbf{V} \in \mathbb{R}^{k \times d}$ is orthonormal as before. Taking the log-likelihood recovers a very similar squared-error criterion, and the maximum likelihood solution for $\mu$ and $\mathbf{V}$ is exactly the PCA solution. Therefore, another interpretation of PCA is that we suppose the data is Gaussian and that we are looking for a subspace that "explains" most of the *variance* in the data. Either way, PCA gives us the best $k$-dimensional representation of the data in terms of squared reconstruction error.

#### 4.2.4.2 $k$-means Clustering

In $k$-means clustering, we are after a rather different type of simplification of the data set. The objective is to learn a function rather similar to a classifier that maps points $\mathbf{x} \in \mathbb{R}^d$ to "labels" $y \in \{1, \ldots, k\}$. In clustering, we use the **within-point scatter** as the loss function, which we define directly over the training set:

$$L_n(f) = \sum_{i=1}^{n} \sum_{j:f(\mathbf{x}_j)=f(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{x}_j\|^2 . \tag{4.153}$$

That is, we want to choose the clustering function such that a point is as close as possible to all of the other points assigned to the same cluster. It is straightforward to see that the empirical loss can be written as

$$L_n(f) = \sum_{i=1}^{k} N_k \sum_{f(\mathbf{x}_j)=k} \|\mathbf{x}_i - \mu_i\|^2 , \tag{4.154}$$

where $N_i$ is the number of points assigned to cluster $i$, and $\mu_i$ is the average value of the points assigned to cluster $k$.

This formulation inspires the $k$-means method: Choose $k$ mean vectors around which each cluster will be centered, and then cluster each point $\mathbf{x}_i$ to the nearest mean. However, this leads to a chicken-and-egg problem: We can't choose the best means until we know which cluster each point is assigned to, but we can't cluster each point until we know the means defining each cluster. In principle, we could try out all possible clusterings of the points and see which one leads to the minimum scatter. In practice, this algorithm has enormous complexity. Solving the clustering problem is NP Hard, so we cannot expect an efficient and exact solution.

Instead, we take an iterative approach. First, choose $k$ mean vectors $\mu_i$ arbitrarily, and cluster all $n$ points $\mathbf{x}_i$ to the nearest mean. Then, carry out the following steps:

1. Given cluster assignments $f(\mathbf{x}_i)$, update the means to be the sample mean of each cluster:

$$\mu_i = \frac{1}{n} \sum_{f(\mathbf{x}_j)=i} \mathbf{x}_j . \tag{4.155}$$

2. Given means $\mu_i$, cluster each point to the nearest mean:

$$f(\mathbf{x}_j) = \arg\min_i \|\mathbf{x}_j - \mu_i\|^2 . \tag{4.156}$$

These steps iterate until the cluster assignments do not change.

$k$-means is guaranteed to converge; updating the mean and choosing better cluster assignments can only decrease the within-class scatter, so eventually the algorithm settles on an answer. However, there is no guarantee that this is a global optimum. In general it will not be, and it is common to initialize the algorithm with multiple choices of means, and choose the clustering that gives the smallest overall scatter.

# Chapter 5

# Bayesian Estimation

In Bayesian estimation, we make the crucial assumption that we have access to a *prior* distribution over the parameter space over which we are trying to estimate. As in detection theory, the existence of the prior is controversial: where to we get the prior, and what does it mean for a prior to be the "true" one? When we can assert a prior distribution, it will greatly facilitate the estimation process. Rather than look for a single unbiased estimator that uniformly minimizes the variance, we will look for estimators that minimize *average* error. Our main target will be the **minimum mean squared error** (MMSE) estimator, which, by contrast to the MVUE estimator, virtually always exists. We will also see that we can use the prior to encode important information about the *structure* of a signal, which substantially improves the performance in the case of signal denoising, inpainting, or deconvolution. An important example of such structure is *sparsity*, in which the signal is modeled as a linear combination of only a few elements of a large basis set in a manner similar to PCA. In modern data science, priors are often learned empirically from datasets. We will investigate a popular method for learning the data model, called $K$-SVD, that finds a basis in which signals admit a sparse representation.

While Bayesian estimation is rather powerful, it can also be difficult to carry out. This is because we usually must compute the *posterior distribution* $p(\theta|y)$ using Bayes rule, which in general involves a challenging integration. This motivates the definition of **conjugate priors**, which are priors $p(\theta)$ such that the posterior $p(\theta|y)$ is tractable. We will find that as long as $p(y|\theta)$ belongs to an exponential family, there exists a conjugate prior $p(\theta)$ that ensures that one can compute the posterior without too much difficulty.

## 5.1 Scalar Parameter Estimation

We will begin with the case in which the parameter to estimate is a scalar. Just as in non-random estimation, we define the **likelihood function**

$$p(y|\theta), y \in \Gamma, \theta \in \Lambda, \tag{5.1}$$

but here we define a **prior distribution** over $\theta$

$$p(\theta), \theta \in \Lambda. \tag{5.2}$$

An **estimator** maps the observation $y$ to an estimate of $\theta$:

$$\hat{\theta}(y) : \Gamma \to \Lambda. \tag{5.3}$$

Finally, we choose a **cost function**

$$C(\hat{\theta}(y), \theta) : \Lambda \times \Lambda \to \mathbb{R}. \tag{5.4}$$

The cost function $C(\cdot, \cdot)$ characterizes the quality of the estimate, similar to the way the bias and variance of an estimator characterized estimate quality in non-random estimation. We will look at three important cases of the cost function shortly.

Each of these objects has its analogy in detection theory. In hypothesis testing, we aimed to infer which of finitely many hypotheses $H_0, H_1, \ldots$ is true from the observation $Y$ and the prior probabilities of the hypotheses. We did this by choosing the detector that minimizes the Bayes risk, or the expected cost. In estimation, we want to infer the value of a parameter $\theta \in \Lambda$. The parameter $\theta$ is usually a continuous random variable, and it may also be a vector, in which case $p(\theta)$ is a *multivariate* probability density function. The observation $Y$ will also usually (but not always!) be a continuous random variable (or vector), in which case $p(y|\theta)$ is a (perhaps multivariate) density function.

We choose the estimator $\hat{\theta}(y)$ to minimize the Bayes risk, which similar to before is the expected cost. In the estimation setting, it has the following form:

$$r(\hat{\theta}) = E[C(\hat{\theta}(y), \theta)] = \int_\Gamma \int_\Lambda C(\hat{\theta}(y), \theta) p(y|\theta) p(\theta) d\theta dy. \tag{5.5}$$

In other words, for every $y$ and $\theta$ we compute the estimate $\hat{\theta}$ and the cost $C(\hat{\theta}(y), \theta)$, and we average the result over the joint density function $p(y, \theta) = p(y|\theta)p(\theta)$.

To describe the Bayes-optimum detector, we need to introduce a new concept: the **conditional expectation**.

**Definition 5.1:.** *For two random variables (or vectors) $X, Y$ having joint distribution $p(x, y)$, the* **conditional expectation** *of $X$ given $Y = y$ is*

$$E[X|Y = y] = \int x p(x|y) dx, \tag{5.6}$$

*which is a function of the random variable $Y$.*

It's easy to see that $E[X] = E[E[X|Y = y]]$, where the outer expectation involves integrating over $Y$:

$$E[X] = E[E[X|Y = y]] = \int y p(y) E[X|Y = y] dy. \tag{5.7}$$

With this definition, we can rewrite the Bayes risk as

$$r(\hat{\theta}) = E[C(\hat{\theta}(y), \theta)] = E[E[C(\hat{\theta}(y), \theta)|Y = y]] = \int p(y) \int C(\hat{\theta}(y), \theta) p(\theta|y) d\theta dy. \tag{5.8}$$

In order to minimize the Bayes risk, we want to choose the detector to minimize the inner integrand. That is, we choose the detector to minimize the *conditional* expected cost:

$$\hat{\theta}(y) = \arg\min_{\hat{\theta}} \int C(\hat{\theta}(y), \theta) p(\theta|y) d\theta = E[C(\hat{\theta}(y), \theta)|Y = y]. \tag{5.9}$$

Choosing the Bayes-optimal detector requires that we compute the *posterior* distribution $p(\theta|y)$ and find the value of $\theta$ that minimizes the conditional cost function. We will look at three different cost functions: squared error, absolute error, and uniform error. They will, in turn, give rise to estimators that choose the conditional mean, median, and mode of $\theta$.

### 5.1.1 Squared Error

When $\Gamma = \mathbb{R}$, by far the most common cost function is the **squared error function**:

$$C(\hat{\theta}(y), \theta) = (\hat{\theta}(y) - \theta)^2. \tag{5.10}$$

This gives rise to the famous *mean-squared error* (MSE) criterion, and we call the Bayes estimate for this cost function the *minimum mean-squared error* (MMSE) detector.

A remarkable fact is that the MMSE detector has a simple form. For scalar $Y$ and $\theta$, the conditional cost for the MSE criterion is

$$\begin{aligned} E[C(\hat{\theta}(y), \theta)|Y = y] &= E[(\hat{\theta}(y) - \theta)^2|Y = y] && (5.11)\\ &= E[\hat{\theta}(y)^2 - 2\theta\hat{\theta}(y) + \theta^2|Y = y] && (5.12)\\ &= \hat{\theta}(y)^2 - 2\hat{\theta}(y)E[\theta|Y = y] + E[\theta^2]. && (5.13) \end{aligned}$$

To minimize the MSE, we simply take the derivative with respect to the estimator $\hat{\theta}(y)$ and set it equal to zero:

$$\begin{aligned} 2\hat{\theta}(y) - 2E[\theta|Y = y] &= 0 && (5.14)\\ \hat{\theta}(y) &= E[\theta|Y = y]. && (5.15) \end{aligned}$$

This means that the MMSE estimate is simply the conditional expectation of $\theta$ given $Y = y$:

$$\hat{\theta}_{MMSE}(y) = E[\theta|Y = y]. \tag{5.16}$$

In order to find the MMSE estimate, "all" we need to do is compute the posterior distribution $p(\theta|y)$ and compute its expectation. In some cases this will be straightforward, and other times we will have to work rather hard to compute the conditional mean.

**Example 5.1: Exponential distribution.** *Let's suppose that the observation $Y^n$ is $n$ i.i.d. samples from an exponential distribution:*

$$p(y_i|\theta) = \begin{cases} \theta \exp(-\theta y), & y \geq 0 \\ 0, & y < 0 \end{cases}. \tag{5.17}$$

*Recall that the exponential distribution models delay intervals between events. For example, the wait time in a queue at a supermarket, or the delay between packets in a communications network, is well modeled by an exponential distribution. The mean of $Y_i$, or the average wait/delay time, is $1/\theta$. Similar to previous examples, the likelihood is the product:*

$$p(y^n|\theta) = \prod_{i=n}^{n} p(y_i|\theta) = \theta^n \exp(-\theta \sum_{i=1}^{n} y_i). \tag{5.18}$$

*We suppose further an exponential prior on $\theta$:*

$$p(\theta) = \begin{cases} \alpha \exp(-\alpha\theta), & \theta \geq 0 \\ 0, & \theta < 0 \end{cases}. \tag{5.19}$$

*This choice looks arbitrary at first blush. One justification is that it "works"; we will see shortly that this choice allows for straightforward computation of the posterior. We will give another justification after we compute the MMSE estimate.*

*To find the MMSE estimate, we first compute the posterior:*

$$p(\theta|y) = \frac{\alpha\theta^n \exp(-\theta(\sum_{i=1}^n y_i + \alpha))}{\int_0^\infty \alpha\theta^n \exp(-\theta(\sum_{i=1}^n y_i + \alpha))d\theta} \tag{5.20}$$

$$= \frac{\theta^n \exp(-\theta(\sum_{i=1}^n y_i + \alpha))}{(\sum_{i=1}^n y_i + \alpha)^{-n-1} n!}. \tag{5.21}$$

*where we integrated by parts to compute the denominator. This density is depicted in Figure 5.1 for several values of $n$. Then, we compute the conditional expectation:*

$$\hat{\theta}_{MMSE}(y) = E[\theta|Y^n = y^n] \tag{5.22}$$

$$= \int_0^\infty \theta p(\theta|Y = y)d\theta \tag{5.23}$$

$$= \frac{(\sum_{i=1}^n + \alpha)^{n+1}}{n!} \int_0^\infty \theta^{n+1} \exp(-\theta(\sum_{i=1}^n y_i + \alpha))d\theta \tag{5.24}$$

$$= \frac{(\sum_{i=1}^n y_i + \alpha)^{n+1}}{n!} \cdot (\sum_{i=1}^n + \alpha)^{-n-2}(n+1)! \tag{5.25}$$

$$= \frac{n+1}{\alpha + \sum_{i=1}^n y_i}, \tag{5.26}$$

*where the integration is again performed by parts. Recall that the ML estimate of $\theta$ was the reciprocal of the sample mean, which is exactly what we get from the MMSE estimator as $\alpha \to \infty$. Indeed, the MMSE estimate "nudges" the ML estimate towards the mean of the posterior, which is $1/\alpha$. This is the sense in which Bayesian estimators are unbiased; they bias the estimate according to the prior distribution. For large $n$, the impact of the prior is negligible, swamped out by the data given by observations. This is a recurring theme in Bayesian estimation. Priors matter the most when data is scarce.*

*The form of the estimate gives another interpretation of the prior. The MMSE estimator is the same form as the ML estimate, but with $n + 1$ measurements, the final being of value $\alpha$! That is, we can look at the prior as a single "virtual" measurement, the impact of which diminishes as $n \to \infty$. Later, when we discuss conjugate priors, we will see how to define "stronger" priors that resemble multiple virtual measurements.*

Note that no MVUE exists for the exponential case, but the MMSE estimator exists and is relatively straightforward to derive. This is because unbiasedness and minimum variance are *uniform* properties that hold for *all* values of $\theta$. This strict requirement is relaxed in Bayesian estimation. All we need is an estimator that is best on the average over $p(\theta)$. The bias of the MMSE estimator is quite high for $\theta$ far away from $1/\alpha$, which is not as important because such values of $\theta$ are unlikely to turn up.

Let's look at an example in which the MVUE does exist: estimation of the Gaussian mean.

**Example 5.2: Gaussian Mean.** *Let's consider the case of estimating the mean of a Gaussian distribution from $n$ i.i.d. samples. In this case, the observation $Y^n$ is a vector $n$ i.i.d. samples from*

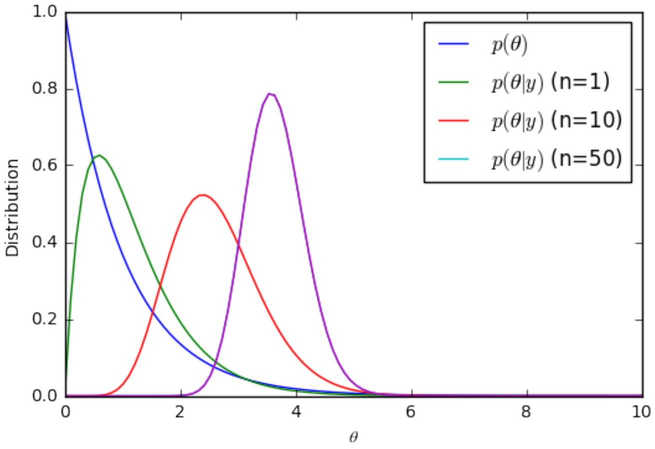

Figure 5.1: Letting the prior be determined by $\alpha = 1$, and supposing the true rate parameter is $\theta = 4$, the posterior $p(\theta|y^n)$ for random samples given $n = 1$, $n = 10$, and $n = 50$. As the number of samples increases, the posterior transitions from the prior distribution to a distribution clustered around the true value of $\theta$.

*a Gaussian distribution, where $\theta$ is the unknown mean of the distribution and $\sigma^2$ is the known variance. Each observation is $Y_i \sim \mathcal{N}(\theta, \sigma^2)$. The likelihood function in this case is*

$$p(y|\theta) = \prod_{i=1}^{n} p(y_i|\theta) \tag{5.27}$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_i - \theta)^2}{2\sigma^2}) \tag{5.28}$$

$$= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \theta)^2). \tag{5.29}$$

*We'll assume that the prior on $\theta$ is Gaussian, too, with zero mean and fixed variance:*

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-\frac{\theta^2}{2\sigma_m^2}). \tag{5.30}$$

*Naively, we might expect the estimator to be the sample average of the observations:*

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{5.31}$$

*However, the MMSE estimate is somewhat more complicated, as it combines information gleaned from the observations with the information contained in the prior.*

*To compute the MMSE estimate, we need to compute the posterior distribution*

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_{-\infty}^{\infty} p(y|\theta)p(\theta)d\theta}. \tag{5.32}$$

*The integral in the denominator will get very unwieldy, so for the moment let's focus on the numerator. The punchline is that the conditional distribution is a Gaussian, so our goal is to show* *that the numerator has exactly the form of a Gaussian distribution. Then, the denominator—which is just a constant in $\theta$—must be whatever value is necessary to make the posterior integrate to one. Knowing this, it will be easy to figure out the constant.*

*We have:*

$$p(y|\theta)p(\theta) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-\frac{\theta}{2\sigma_m^2}) \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\theta)^2) \tag{5.33}$$

$$= \frac{1}{\sqrt{2\pi\sigma_m^2}(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n} y_i^2 + n\theta^2 - 2\theta\sum_{i=1}^{n} y_i\right) - \frac{\theta^2}{2\sigma_m^2}). \tag{5.34}$$

*Now, let's collect all of the terms that do not depend on $\theta$ into a single expression. Define*

$$C(y) = \frac{1}{\sqrt{2\pi\sigma_m^2}(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} y_i^2). \tag{5.35}$$

*Then, we can rewrite the numerator as*

$$p(y|\theta)p(\theta) = C(y)\exp(-\left(\frac{n\theta^2}{2\sigma^2} - \frac{2n\theta\bar{y}}{2\sigma^2} + \frac{\theta^2}{2\sigma_m^2}\right)) \tag{5.36}$$

$$= C(y)\exp\left(-\left(\left(\frac{n}{2\sigma^2} + \frac{1}{2\sigma_m^2}\right)\theta^2 - \frac{2n\bar{y}}{2\sigma^2}\theta\right)\right). \tag{5.37}$$

*If we define*

$$\sigma_{\theta|y}^2 = \frac{1}{\frac{n}{\sigma_2} + \frac{1}{\sigma_m^2}} \tag{5.38}$$

$$\mu_{\theta|y} = \frac{n\sigma_{\theta|y}^2}{\sigma^2}\bar{y}, \tag{5.39}$$

*we can rewrite the numerator as*

$$p(y|\theta)p(\theta) = C(y)\exp\left(-\frac{1}{2\sigma_{\theta|y}^2}(\theta - \mu_{\theta|y})^2\right)\exp(\frac{\mu_{\theta|y}^2}{\sigma_{\theta|y}^2}). \tag{5.40}$$

*The final exponential is a constant in $\theta$, whereas the first exponential is precisely the exponential term for a Gaussian distribution with mean $\mu_{\theta|y}$ and variance $\sigma_{\theta|y}^2$. Therefore, because the posterior must integrate to one, we conclude that*

$$p(\theta|y) = \frac{1}{\sqrt{2\pi\sigma_{\theta|y}^2}}\exp\left(-\frac{(\theta-\mu_{\theta|y})^2}{2\sigma_{\theta|y}^2}\right). \tag{5.41}$$

*Once we know that the conditional distribution is Gaussian, and recalling that the MMSE estimator is the conditional mean of the posterior, we finally get*

$$\hat{\theta}_{MMSE}(y) = \mu_{\theta|y} = \frac{\sigma_m^2}{\sigma_m^2 + \sigma^2/n}\bar{y}. \tag{5.42}$$

*In other words, as $n$ becomes large relative to the noise variance, the MMSE estimator trusts the sample average $\hat{y}$. However, if the number of samples is small, the estimator "shrinks" its answer towards the origin. This is because* a priori *we supposed that $\theta$ had zero mean. In the absence of much data, we make an estimate closer to that mean.*

### 5.1.2 Absolute Error

Another cost function commonly used when $\Gamma = \mathbb{R}$ is the **absolute error function**:

$$C(\hat{\theta}(y), \theta) = |\hat{\theta}(y) - \theta|. \tag{5.43}$$

Instead of penalizing the squared error of the estimator, we penalize its absolute value. We call the resulting estimator $\hat{\theta}(y)$ the *minimum-mean absolute error* (MMAE) estimator.

As with the MMSE estimator, the MMAE estimator has a simple form, this time the conditional *median* instead of the conditional mean. The derivation for this result is a little more complicated, but the resulting estimator $\hat{\theta}_{ABS}(y)$ is the one that satisfies the following:

$$Pr(\theta < \hat{\theta}_{ABS}(y)|Y = y) = 1/2 \tag{5.44}$$

$$\int_{-\infty}^{\hat{\theta}_{ABS}(y)} p(\theta|y)d\theta = \int_{\hat{\theta}_{ABS}(y)}^{\infty} p(\theta|y)d\theta = 1/2. \tag{5.45}$$

as long as the posterior distribution $p(\theta|y)$ is smooth. All this means is that half of the posterior probability is below $\hat{\theta}_{ABS}(y)$, and half of the probability is greater than it, which is exactly the definition of the median. We will concisely represent this estimator with the following notation:

$$\hat{\theta}_{ABS}(y) = \text{median}(\theta|Y = y), \tag{5.46}$$

where the RHS just refers to the conditional median.

One might ask what difference it makes to use the median instead of the mean. Let's look at this in view of the exponential example we considered in the MMSE case.

**Example 5.3: Exponential Distribution.** *Consider the same exponential observation with an exponential prior. For simplicity, we'll suppose we get only a single observation, so $n = 1$. In order to compute the MMAE estimate, we look at the posterior*

$$p(\theta|y) = (y + \alpha)^2 \theta \exp(-\theta(y + \alpha)). \tag{5.47}$$

*The MMAE estimate is the one that satisfies*

$$\int_{\hat{\theta}_{ABS}(y)}^{\infty} (y + \alpha)^2 \theta \exp(-\theta(y + \alpha))d\theta = 1/2 \tag{5.48}$$

$$[1 + (y + \alpha)\hat{\theta}_{ABS}(y)] \exp(-(y + \alpha)\hat{\theta}_{ABS}(y)) = 1/2. \tag{5.49}$$

*It turns out that we cannot solve for $\hat{\theta}_{ABS}(y)$ in closed form. Numerically, we get*

$$\hat{\theta}_{ABS}(y) = \frac{T_0}{\alpha + y}, \tag{5.50}$$

*where $T_0$ is the solution to $(1 + T_0)\exp(-T_0) = 1/2$, which gives $T_0 \approx 1.68$.*

*The MMAE estimator has the same form as the MMSE estimator, except that the constant in the numerator is smaller. The MMAE estimator compensates for the heavy-tailed exponential distribution by "shrinking" the estimate towards zero.*

The MMAE estimate is less sensitive to a "heavy tail" in the posterior distribution, or to "bumps" far away from the center of the distribution. This results in an estimate that is, among other things, robust to outliers.

**Example 5.4: Gaussian Mean.** *Let's again consider the case of estimating the mean of a Gaussian distribution from $n$ i.i.d. samples. Again let*

$$p(y|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \theta)^2), \tag{5.51}$$

*and*

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-\frac{\theta}{2\sigma_m^2}). \tag{5.52}$$

*Here, we need to compute the median of the posterior distribution. We saw previously that, the posterior is*

$$p(\theta|y) = \mathcal{N}(\mu_{\theta|y}, \sigma^2_{\theta|y}), \tag{5.53}$$

*with the mean and variance as defined above. We can exploit a nice fact about the Gaussian: the mean and the median are the same! Half of the probability mass lies above the mean, and half of it lies below it. So, in this case, the MMAE estimator is identical to the MMSE estimator:*

$$\hat{\theta}_{MMAE}(y) = \hat{\theta}_{MMSE}(y) = \frac{\sigma_m^2}{\sigma_m^2 + \sigma^2/n} \bar{y}. \tag{5.54}$$

### 5.1.3 Uniform Error, or Maximum *a Priori* Estimation

Another popular choice when $\Gamma = \mathbb{R}$ is the **uniform error function**:

$$C(\hat{\theta}(y), \theta) = \begin{cases} 0, & |\hat{\theta}(y) - \theta| \leq \Delta \\ 1, & |\hat{\theta}(y) - \theta| > \Delta, \end{cases} \tag{5.55}$$

for some $\Delta > 0$, usually taken to be a small number. In other words, if the estimate is within a small radius of the true value of $\theta$, we suffer no cost, and otherwise we suffer a uniform cost. Once the estimate veers more that $\Delta$ away from the true value, we aren't interested in knowing how close the estimate is; we simply consider the estimation a failure and assign unit cost to it.

Uniform error leads to an extremely important estimator: the *maximum a priori* (MAP) estimator. To derive it, let's look at the Bayes risk of an estimator:

$$E[C(\hat{\theta}(y), \theta)] = 1 - \int_{\hat{\theta}(y)-\Delta}^{\hat{\theta}(y)+\Delta} p(\theta|y) d\theta \tag{5.56}$$

$$\approx 1 - 2\Delta p(\theta|y)_{\theta=\hat{\theta}(y)}, \tag{5.57}$$

where the approximation holds when $\Delta$ is small. In order to make the Bayes risk as small as possible, we want to choose $\hat{\theta}(y)$ in order to make the posterior as large as possible. Therefore, we choose the estimator

$$\hat{\theta}_{MAP}(y) = \arg\max_{\theta} p(\theta|y). \tag{5.58}$$

A useful property of the MAP estimator is that it is relatively easy to compute. This is because we do not need to calculate the posterior exactly in order to find $\hat{\theta}_{MAP}(y)$. To see this, recall that we compute the posterior via Bayes rule on the likelihood and prior:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int_\Gamma p(y|\theta)p(\theta)d\theta}. \tag{5.59}$$

The integral in the denominator is often challenging to compute. Fortunately, we do not need to know it! It does not depend on $\theta$, so once we have observed $Y$, the MAP estimate is equivalently

$$\hat{\theta}_{MAP}(y) = \arg\max_{\theta} p(y|\theta)p(\theta). \tag{5.60}$$

**Example 5.5: Exponential distribution.** *To compute the MAP estimator, we find it easier first to take the logarithm of the (unnormalized) posterior*

$$\hat{\theta}_{MAP}(y^n) = \arg\max_{\theta} \log(p(y^n|\theta)) + \log(p(\theta)) \tag{5.61}$$

$$= \arg\max n\log(\theta) - \theta\sum_{i=1}^{n} y_i + \log(\alpha) - \theta\alpha. \tag{5.62}$$

*To maximize, we take the derivative and set it equal to zero:*

$$n/\theta - (\alpha + \sum_{i=1}^{n} y_i) = 0 \tag{5.63}$$

$$\theta = \frac{n}{\alpha + \sum_{i=1}^{n} y_i}. \tag{5.64}$$

*So, we arrive at the estimate*

$$\hat{\theta}_{MAP}(y^n) = \frac{n}{\alpha + \sum_{i=1}^{n} y_i}. \tag{5.65}$$

*Again, the estimator has the same form as the MMSE estimator, but the MAP estimator shrinks even closer to the origin. The posterior is rather heavy-tailed, so its mode is closer to the origin than its mean.*

**Example 5.6: A Biased Coin.** *Suppose that we observe $n$ i.i.d. flips of a coin with the probability of heads equal to $0 \le \theta \le 1$. The likelihood function of the observation $Y^n$ is:*

$$p(y^n|\theta) = \theta^{\sum_{i=1}^{n} y_i}(1-\theta)^{n-\sum_{i=1}^{n} y_i}. \tag{5.66}$$

*Let's suppose a uniform prior over the bias of the coin:*

$$p(\theta) = \begin{cases} 1, & 0 \le \theta \le 1 \\ 0, & otherwise \end{cases}. \tag{5.67}$$

*To compute the MAP estimator, we need to maximize the product $p(y^n|\theta)p(\theta)$:*

$$\hat{\theta}_{MAP}(y) = \arg\max_{\theta} \log(p(y^n|\theta)) + \log(p(\theta)) \tag{5.68}$$

$$= \arg\max_{\theta} + \log(\theta)\sum_{i=1} y_i + \log(1-\theta)(n - \sum_{i=1} y_i), \tag{5.69}$$

*where the effect of the prior vanishes because it is equal to one. Setting the derivative equal to zero, we get*

$$\frac{\sum_{i=1} y_i}{\theta} + \frac{n - \sum_{i=1} y_i}{1-\theta} = 0 \tag{5.70}$$

$$\theta = \frac{1}{n}\sum_{i=1} y_i. \tag{5.71}$$

*So,*

$$\hat{\theta}_{MAP}(y) = \frac{1}{n}\sum_{i=1} y_i, \tag{5.72}$$

*which is exactly the sample mean estimator.*

Note that in this case, the prior did not impact the MAP estimate. Whenever the prior is uniform over $\theta$, the MAP estimator is the maximum-likelihood estimator. Furthermore, when the prior is *approaching* a uniform distribution, such as a Gaussian prior with large variance, the MAP estimator is *approximately* equal to the one maximizing the likelihood function.

**Example 5.7: Gaussian Mean.** *Let's look one more time at estimating a Gaussian mean, where we have*

$$p(y|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \theta)^2), \tag{5.73}$$

*and*

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-\frac{\theta^2}{2\sigma_m^2}). \tag{5.74}$$

*Again the posterior is*

$$p(\theta|y) = \mathcal{N}(\mu_{\theta|y}, \sigma_{\theta|y}^2), \tag{5.75}$$

*with the mean and variance as defined above. Once again we have a nice fact: the mode of a Gaussian is its mean. So, the MAP estimator is identical to the MMSE estimator:*

$$\hat{\theta}_{MAP}(y) = \hat{\theta}_{MMAE}(y) = \hat{\theta}_{MMSE}(y) = \frac{\sigma_m^2}{\sigma_m^2 + \sigma^2/n}\bar{y}. \tag{5.76}$$

*This is a nice feature of Gaussian estimation. Regardless of whether we want to minimize the squared, absolute, or uniform error, the optimum estimator is just the conditional mean.*

### 5.1.4 Vector Parameter Estimation

So far we have looked only at cases in which $\theta$ is a single parameter, like a mean or a Bernoulli parameter. But there are plenty of cases in which we want to estimate a *vector* $\theta$, such as when there is a multivariate Gaussian whose mean we want to estimate. In this case, we need to define a cost function $C(\hat{\theta}, \theta)$ that makes sense when $\Lambda = \mathbb{R}^d$ for some $d > 1$.

#### 5.1.4.1 Vector MMSE estimation

By analogy with the scalar case, we'll consider three vector-valued functions that extend the squared, absolute, and uniform errors. Let's start with the **squared Euclidean error**:

$$C(\hat{\theta}(y), \theta) = \left\|\hat{\theta}(y) - \theta\right\|^2, \tag{5.77}$$

where $\|\cdot\|^2$ is the squared Euclidean norm:

$$\|x\|^2 = \sum_{i=1}^{d} x_i^2. \tag{5.78}$$

We still call the estimator that minimizes the average Euclidean squared error the MMSE estimator, since the error is still squared error in each element of the vector parameter $\theta$. To find the MMSE estimator, we can apply the same argument as we did in the vector case to each element of the vector estimate $\hat{\theta}(y)$. This shows that the MMSE estimator is still the conditional expectation:

$$\theta_{MMSE}(y) = E[\theta | Y = y], \tag{5.79}$$

where this is a *vector* mean instead of just a scalar.

#### 5.1.4.2 Vector MMAE estimation

We can also define an analog of the scalar absolute error. Letting $\theta = (\theta_i, \ldots, \theta_d)$, we can write this cost as

$$C(\hat{\theta}(y), \theta) = \sum_{i=1}^{d} |\hat{\theta}_i - \theta_i| = \left\| \hat{\theta} - \theta \right\|_1, \tag{5.80}$$

where $\|\cdot\|_1$ is called the $\ell_1$ norm. Just like before, we penalize the magnitude of the deviation from the estimate, and here we sum up the deviation over all of the elements of the parameter vector.

Here, too we can repeat the scalar argument for every element of $\theta$, and as expected we get the conditional median as the MMAE estimator:

$$\hat{\theta}_{ABS}(y) = \text{median}(\theta | Y = y), \tag{5.81}$$

where the median is computed element-wise.

#### 5.1.4.3 Vector Uniform Cost

In order to generalize the uniform cost, we have to work a little harder. Instead of considering each element of $\theta$ individually, we define a cost function that considers the worst-case deviation from $\theta$:

$$C(\hat{\theta}_i(y), \theta) = \begin{cases} 0, & \max_{1 \le i \le d} |\hat{\theta}_i - \theta_i| \le \Delta \\ 1, & \max_{1 \le i \le d} |\hat{\theta}_i - \theta_i| > \Delta \end{cases}. \tag{5.82}$$

That is, if any element of the estimate $\hat{\theta}$ differs from $\theta$ by more than $\Delta$, then we suffer a cost, and otherwise the cost is zero. If $\Delta$ is small, we again can approximate the expected value of $C$ by 1 minus the posterior, which leads to the MAP estimator:

$$\hat{\theta}_{MAP}(y) = \arg\max_{\theta} p(\theta | y) = \arg\max_{\theta} p(y|\theta) p(\theta). \tag{5.83}$$

Once again, in the Gaussian case the three estimates coincide. Let's look at a very important Gaussian estimation problem: estimating one Gaussian vector from a look at a correlated vector.

**Example 5.8: Correlated Gaussians.** *Let $\Lambda = \mathbb{R}^d$ and $\Gamma = \mathbb{R}^n$. Both $y$ and $\theta$ are Gaussian*

$$Y \sim \mathcal{N}(\mu_y, \Sigma_y) \tag{5.84}$$

$$\theta \sim \mathcal{N}(\mu_\theta, \Sigma_\theta), \tag{5.85}$$

*for mean vectors $\mu_y \in \mathbb{R}^n$, $\mu_\theta \in \mathbb{R}^d$ and covariance matrices $\Sigma_y \in \mathbb{R}^{n\times n}$, $\Sigma_\theta \in \mathbb{R}^{d\times d}$. Furthermore, the two random vectors are jointly Gaussian, with covariance*

$$\Sigma_{\theta y} = E[(\theta - \mu_\theta)(y - \mu_y)^T] \in \mathbb{R}^{d\times n}. \tag{5.86}$$

*It's a straightforward but tedious process to derive the conditional distribution:*

$$\theta | Y = \mathcal{N}(\mu_\theta + \Sigma_{\theta y}\Sigma_y^{-1}(y - \mu_y), \Sigma_\theta - \Sigma_{\theta y}\Sigma_y^{-1}\Sigma_{\theta y}^T). \tag{5.87}$$

*The MMSE (and MMAE and MAP) estimate is therefore*

$$\mu_\theta + \Sigma_{\theta y}\Sigma_y^{-1}(y - \mu_y). \tag{5.88}$$

## 5.2 Conjugate Priors

The major challenge to Bayesian estimation is computing the posterior $p(\theta|y)$, which often involves a challenging integration. In order to keep Bayesian estimation tractable, Bayesian statisticians have developed the notion of **conjugate priors**: families of prior distributions $p(\theta)$ that are matched to families of likelihood functions $p(y|\theta)$ such that the posterior distribution $p(\theta|y)$ has a closed-form expression. We'll first give a formal definition, and then we'll look at several examples.

**Definition 5.2: Conjugate Priors.** *Let $\mathcal{F}$ be a family of likelihood functions $p(y|\theta)$ described by parameters $\theta$, and let $\mathcal{P}$ be a family of prior distributions $p(\theta)$. We say that the family $\mathcal{P}$ is* **conjugate** *to $\mathcal{F}$ if*

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \in \mathcal{P}, \tag{5.89}$$

*for every likelihood in $\mathcal{F}$ and every prior in $\mathcal{P}$.*

In other words, when we use a conjugate prior, the posterior is in the *same family* as the prior. Usually, this means that the posterior is rather easy to evaluate, often involving simple calculations of the data applied to the prior.

An extremely useful result is that exponential families are conjugate to each other. If the likelihood is a member of the exponential family, then there is a family of exponential priors such that the posterior, too, is a form of the exponential family. This is an extremely useful fact, since exponential families are so common in estimation and learning problems. For the rest of this section, we will investigate examples of conjugate families and their use in estimation and learning problems.

### 5.2.1 Gaussian Families

Let $\mathcal{F}$ be the family of (multivariate) Gaussian distributions $p(\mathbf{y}|\mu)$, and suppose we obtain $n$ i.i.d. samples drawn from this distribution $\mathbf{y}_i$. How can we perform Bayesian estimation on $\mu$?

In this case, the conjugate prior, too, is Gaussian. We suppose that the mean is *a priori* a Gaussian random variable:

$$\mu \sim \mathcal{N}(\mu_0, \Sigma_0) = p(\mu). \tag{5.90}$$

The parameters $\mu_0$ and $\Sigma_0$ are often called the **hyperparameters** of the prior distribution, and they specify the strength of our prior beliefs on $\mu$. Given a set of observations $\mathbf{y}_i$, we can compute the posterior

$$p(\mu|\mathbf{y}^n) = \frac{p(\mu)\prod_{i=1}^n p(\mathbf{y}_i|\mu)}{\int p(\mu)\prod_{i=1}^n p(\mathbf{y}_i|\mu)d\mu}. \tag{5.91}$$

Similar to the case of estimating a scalar Gaussian, it is possible to see that the numerator of the preceding expression is the exponential of a quadratic form, which proves that the posterior is also a Gaussian. In particular, algebraic manipulations show that the mean and covariance of the posterior are

$$\mu_{\theta|y} = (\Sigma_0^{-1} + n\Sigma^{-1})^{-1}(\Sigma_0^{-1}\mu_0 + n\Sigma^{-1}\bar{\mathbf{y}}) \tag{5.92}$$

$$\Sigma_{\theta|y} = (\Sigma_0^{-1} + n\Sigma^{-1})^{-1}. \tag{5.93}$$

Thus, the MMSE (and MAP) estimate of the mean is

$$\hat{\mu} = (\Sigma_0^{-1} + n\Sigma^{-1})^{-1}(\Sigma_0^{-1}\mu_0 + n\Sigma^{-1}\bar{\mathbf{y}}). \tag{5.94}$$

This is comparable to the ML/MVUE estimate, except biased towards the prior mean $\mu_0$. The degree of this bias depends on the relationship between the strength of the prior, the number of samples $n$, and the strength of the covariance $\Sigma$ on the observations. As $n$ becomes large, the effects of the posterior vanish, and the estimate converges on the sample mean.

Note that we have supposed that the covariance matrix is known. It is also possible to define a conjugate prior over $\Sigma$ and to estimate jointly the mean and covariance. The answer gets a bit more complicated, involving a new distribution called the **Wishart distribution**, and we will skip over it. Be aware that covariance estimation is an important and still-active area of research in statistics and signal processing.

### 5.2.2 The Bernoulli Distribution

Consider a random variable with a Bernoulli distribution:

$$p(y|\theta) = \theta^y(1-\theta)^{1-y}. \tag{5.95}$$

This distribution also is a member of the exponential family, although it may not be as obvious. Therefore, it has a conjugate prior. Its conjugate prior is a new distribution family, called the **Beta distribution**:

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{5.96}$$

where the hyperparameters are $\alpha, \beta > 0$, and where $\Gamma(\cdot)$ is the Gamma function:

$$\Gamma(x) = \int_0^\infty x^{z-1}e^x dz, \tag{5.97}$$

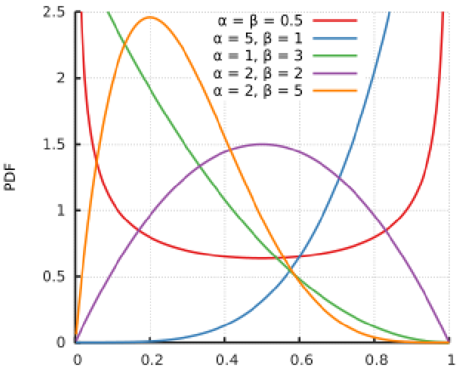

Figure 5.2: The beta distribution. Source: Wikimedia Commons.

which generalizes the factorial to non-integer values. This prior has a similar form to the likelihood function, which gives us an intuitive sense that the posterior will also have the same form. The parameters $\alpha$ and $\beta$ allow one to control the shape of the prior. Several values are shown in Figure

5.2. In a moment we will give further interpretation to these parameters. The mean and mode of the Beta distribution are

$$E[\theta] = \frac{\alpha}{\alpha+\beta} \tag{5.98}$$

$$\arg\max p(\theta) = \frac{\alpha-1}{\alpha+\beta-2}, \tag{5.99}$$

where the expression for the mode holds if $\alpha, \beta > 1$. Otherwise the mode is either 1 or 0.

Given $n$ i.i.d. measurements, we can find the posterior by looking only at the numerator of Bayes rule, and by ignoring normalizing constants:

$$\begin{aligned} p(\theta|y^n) &\propto p(\theta)\prod_{i=1}^{n} p(y_i|\theta) && (5.100) \\ &\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{\sum_{i=1}^{n} y_i}(1-\theta)^{n-\sum_{i=1}^{n} y_i} && (5.101) \\ &\propto \theta^{\alpha+\bar{y}-1}(1-\theta)^{\beta+n-\bar{y}-1}, && (5.102) \end{aligned}$$

where $\bar{y} = \sum_{i=1}^{n} y_i$. Now we can see that the posterior is in the form of a Beta distribution, with parameters

$$\alpha_{\theta|y} = \alpha + \bar{y} \tag{5.103}$$

$$\beta_{\theta|y} = \beta + n - \bar{y}. \tag{5.104}$$

So, the MMSE estimate is the mean of this Beta distribution, or

$$\hat{\theta}_{MMSE} = \frac{\alpha+\bar{y}}{\alpha+\beta+n}. \tag{5.105}$$

Compare this estimator to the ML estimator $\hat{\theta}_{ML} = \bar{y}/n$. Here, we can regard $\alpha$ and $\beta$ as virtual measurements or “pseudocounts”. The MMSE estimator is as though we have seen $\alpha$ instances of $Y = 1$ and $\beta$ instances of $Y = 0$ prior to observing the dataset. Hence we add $\alpha$ measurements to the numerator and $\alpha + \beta$ total measurements to the normalizing constant. The larger we make $\alpha$ and $\beta$, the more virtual measurements our prior asserts, and the stronger the prior. Note that $\alpha$ and $\beta$ need not be integers; we can choose a prior that encodes a fractional number of pseudocounts!

The MAP estimator is the mode of the Beta posterior distribution, or

$$\hat{\theta}_{MAP} = \frac{\alpha+\bar{y}-1}{\alpha+\beta+n-2}. \tag{5.106}$$

The MAP estimator has a similar intuition. The parameters $\alpha$ and $\beta$ perturb the posterior as though there are observations of $Y = 0$ or $Y = 1$ as before. As always, as $n \to \infty$ the MMSE, MAP, and ML estimates coincide.

### 5.2.3 Multinomial Distribution

Consider a random variable with a multinomial distribution:

$$p(\mathbf{y}|\theta) = \frac{n!}{y_1!y_2!\cdots y_k!}\theta_1^{y_1}\theta_2^{y_2}\cdots\theta_k^{y_k}, \tag{5.107}$$

where $k$ is the number of observable events, $y_i$ is the number of times that event $i$ is observed, and $\theta_i$ is the probability that event $i$ occurs. That is, the multinomial distribution models $n$ i.i.d. draws of a discrete random variable that can take on $k$ different values, and the elements $y_i$ count up the number of times we observe each value. Naturally, $\sum_{i=1}^k \theta_i = 1$ and $\sum_{i=1}^k y_i = n$.

The multinomial distribution generalizes the binomial distribution to more than two observable values. It, too, is a member of the exponential family, and we again can define a conjugate prior. In this case, the conjugate prior is yet another new distribution family, the **Dirichlet distribution**:

$$p(\theta) = \frac{1}{B(\theta)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}, \tag{5.108}$$

where each $\alpha_i > 0$, and where $\theta$ must be a valid probability distribution, i.e. $\theta_i \geq 0$ and $\sum_{i=1}^k \theta_i = 1$, and where $B(\theta)$ is called the **multivariate Beta function** and normalizes the distribution:

$$B(\theta) = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^k)}. \tag{5.109}$$

The Dirichlet distribution is a "distribution over distributions"—it gives the probability that the probability distribution of a random variable is equal to $\theta$. Note that the form of this distribution is similar both to the multinomial distribution and to the Beta distribution defined in the previous subsection.

The mean and mode of the Dirichlet distribution are

$$E[\theta_i] = \frac{\alpha_i}{\sum_{i=1}^k \alpha_i} \tag{5.110}$$

$$\arg\max_{\theta_i} = \frac{\alpha_i - 1}{\sum_{i=1}^k \alpha_i - k}, \tag{5.111}$$

where the expression for the mode is valid when each $\alpha_i > 1$.

Estimating the parameters of a multinomial distribution is equivalent to estimating the pmf of a discrete distribution. The observation $\mathbf{y}$ is just the histogram of the observation—how many times did we observe each of the $k$ values? Indeed, the maximum likelihood estimate of $\theta$ is just the normalized histogram, or $\hat{\theta}_{ML} = \mathbf{y}/n$.

To compute the Bayes estimate, we compute the posterior. Similar to the binomial case, we compute the posterior by looking at $p(y|\theta)p(y)$, neglecting the constant terms, and observing the form of the result.

$$p(\theta|\mathbf{y}) \propto p(\mathbf{y}|\theta)p(\theta) \tag{5.112}$$

$$\propto \left(\prod_{i=1}^{k} \theta_i^{y_i}\right)\left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1}\right) \tag{5.113}$$

$$\propto \left(\prod_{i=1}^{k} \theta_i^{\alpha_i + y_i - 1}\right). \tag{5.114}$$

Therefore, we conclude that the posterior distribution is also a Dirichlet distribution:

$$p(\theta|\mathbf{y}) = \frac{1}{B(\alpha + \mathbf{y})} \prod_{i=1}^{k} \theta_i^{\alpha_i + n - 1}. \tag{5.115}$$

The MMSE estimate is therefore

$$\hat{\theta}_{MMSE} = \frac{\alpha + y}{\sum_{i=1}^{k} \alpha_i + n}, \tag{5.116}$$

and the MAP estimate is

$$\hat{\theta}_{MAP} = \frac{\alpha + y - 1}{\sum_{i=1}^{k} \alpha_i + n - 1}. \tag{5.117}$$

Once again, the prior has an interpretation in terms of "pseudocounts". The MMSE estimate of $\theta$ is just the normalized histogram indicated by $\mathbf{y}$, with pseudocounts $\alpha$ added to the histogram. The more counts we add, the stronger the prior, and the longer it takes for measurements to swamp out the impact of the prior. The MAP estimate has a similar intuition, except that the pseudocounts are decremented by one in making the estimate. As $n \to \infty$, all of the estimates converge on the normalized histogram.

**Example 5.9: Spam Detection.** *Recall the spam detection example from Chapter 3, where we applied a bag of words model to the question of whether an email message is spam or normal. For a message of length $n$, and given a dictionary of $k$ words, we supposed a multinomial model for the word counts in each message:*

$$p(\mathbf{y}|H = 0) = \frac{n!}{y_1! \cdots y_k!} (p_1^0)^{y_1} \cdot (p_2^0)^{y_2} \cdots (p_k^0)^{y_k} \tag{5.118}$$

$$p(\mathbf{y}|H = 1) = \frac{n!}{y_1! \cdots y_k!} (p_1^1)^{y_1} \cdot (p_2^1)^{y_2} \cdots (p_k^1)^{y_k}. \tag{5.119}$$

*There, we supposed that the word frequency vectors $\mathbf{p}_0$ and $\mathbf{p}_1$ were known. How might we estimate them from data? Given a large corpus of spam and regular email messages, we can frame this as a Bayesian estimation problem and train a classifier.*

*Suppose that we place Dirichlet priors on $\mathbf{p}_0$ and $\mathbf{p}_1$ with parameters $\alpha_0$ and $\alpha_1$ specifying the pseudocounts. Then, suppose that we have a training set that has multiple normal messages with $n_0$ total dictionary words with word counts $\mathbf{y}_0$, and multiple spam messages with $n_1$ total dictionary words with word counts $\mathbf{y}_1$. Then, we can compute the MMSE estimate of the frequency vectors:*

$$\hat{\mathbf{p}}_0 = \frac{\alpha_0 + \mathbf{y}_0}{\sum_i \alpha_{0i} + n_0} \tag{5.120}$$

$$\hat{\mathbf{p}}_1 = \frac{\alpha_1 + \mathbf{y}_1}{\sum_i \alpha_{1i} + n_1}. \tag{5.121}$$

*Then, we can use these frequency vectors in the spam classifier.*

## 5.3 Bayesian Estimation of Signals in Noise

In this section we will focus on a particular signal model, which we have seen several times before:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}, \tag{5.122}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the observed signal, $\mathbf{x} \in \mathbb{R}^m$ is the signal we hope to estimate and $\mathbf{H} \in \mathbb{R}^{n \times m}$ is the **channel** relating the signal and the observation, and $\mathbf{z}$ is additive noise. In the case of Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, recall that the maximum-likelihood estimate of $\mathbf{x}$ involves the pseudo-inverse of $\mathbf{H}$:

$$\hat{\mathbf{x}}_{ML} = \mathbf{H}^{+}\mathbf{y}. \tag{5.123}$$

If $\mathbf{H}$ is either invertible or "close" to invertible, the ML solution provides acceptable performance. For example, we can use it to sharpen a blurred image.

However, for more severe distortions of the signal, the ML solution is not satisfactory. In the case of inpainting, where $\mathbf{H}$ simply removes entire pixels from the image, the ML solution is no help at all. Similarly, if $m < n$, we saw that the ML solution does a very poor job reconstructing the image. The reason for poor performance in these situations is that the ML solution places no assumptions on the target signal $\mathbf{x}$; it does not "know", for example, that the signal is an image that carries a predictable structure. As a result, it has no way of filling in missing pixels in a corrupted image.

To do better, we will need to encode information about the signal into the estimation scheme. We will do this by asserting a prior distribution $p(\mathbf{x})$ and computing a MMSE or MAP estimate. Depending on how well the prior describes the inherent structure of the signal, we will find that the estimation performance is improved dramatically.

### 5.3.1 General Solution: Gaussian Priors

Let's consider the case where $p(\mathbf{x})$ is Gaussian and the noise is zero-mean Gaussian but otherwise constrained:

$$p(\mathbf{x}) = \mathcal{N}(\mu_x, \Sigma_x), \quad p(\mathbf{z}) = \mathcal{N}(0, \Sigma_z), \tag{5.124}$$

where the mean and covariances are of the correct size—$\mu_x \in \mathbb{R}^m, \Sigma_x \in \mathbb{R}^{m\times m}$, and $\Sigma_z \in \mathbb{R}^{n\times n}$.

To find the MMSE(/MAP) solution, we need only to figure out the overall covariance structure of $\mathbf{x}$ and $\mathbf{y}$. The random vectors $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian, and we can represent their joint distribution via their mean and covariance matrices. First, we solve for the mean.

$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} = \begin{bmatrix} E[\mathbf{x}] \\ E[\mathbf{y}] \end{bmatrix}. \tag{5.125}$$

The mean of $\mathbf{x}$ is given by the prior mean $\mathbf{x}$. The mean of $\mathbf{y}$ is easily determined from the measurement model:

$$\mu_y = E[\mathbf{y}] = E[\mathbf{H}\mathbf{x} + \mathbf{z}] = \mathbf{H}E[\mathrm{x}] + E[\mathbf{z}] = \mathbf{H}\mu_x. \tag{5.126}$$

Next, we find the joint covariance, defined as

$$\Sigma = \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{bmatrix} = \begin{bmatrix} E[(\mathbf{x}-\mu_x)(\mathbf{x}-\mu_x)^T] & E[(\mathbf{x}-\mu_x)(\mathbf{y}-\mu_y)^T] \\ E[(\mathbf{y}-\mu_y)(\mathbf{x}-\mu_x)^T] & E[(\mathbf{y}-\mu_y)(\mathbf{y}-\mu_y)^T] \end{bmatrix}. \tag{5.127}$$

The first block of the covariance is just $\Sigma_x$. The final block we can calculate fairly easily:

$$\begin{aligned}
\Sigma_y = E[(\mathbf{y}-\mu_y)(\mathbf{y}-\mu_y)^T] &= E[(\mathbf{H}(\mathbf{x}-\mu_x)+\mathbf{z})((\mathbf{H}(\mathbf{x}-\mu_x)+\mathbf{z}))^T] && (5.128)\\
&= \mathbf{H}E[(\mathbf{x}-\mu_x)(\mathbf{x}-\mu_x)^T]\mathbf{H}^T + E[\mathbf{z}\mathbf{z}^T] && (5.129)\\
&= \mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z, && (5.130)
\end{aligned}$$

where the cross-terms vanish because the noise is independent of the signal. To calculate the second and third block, note that $\Sigma_{xy} = \Sigma_{yx}^T$ by definition, and

$$\begin{aligned}\Sigma_{xy} &= E[(\mathbf{x} - \mu_x)(\mathbf{y} - \mu_y)^T] && (5.131)\\ &= E[(\mathbf{x} - \mu_x)(\mathbf{H}(\mathbf{x} - \mu_x) + \mathbf{z})^T] && (5.132)\\ &= E[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)^T]\mathbf{H}^T && (5.133)\\ &= \Sigma_x \mathbf{H}^T, && (5.134)\end{aligned}$$

where again the cross-terms vanish because $\mathbf{z}$ is independent of $\mathbf{x}$. Putting all this togther, we get that the joint distribution of $\mathbf{x}$ and $\mathbf{y}$ is

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\mu_x \\ \mathbf{H}\mu_x\end{bmatrix}, \begin{bmatrix}\Sigma_x & \Sigma_x \mathbf{H}^T \\ \mathbf{H}\Sigma_x & \mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z\end{bmatrix}\right). \qquad (5.135)$$

Using this information, we can find the posterior distribution of $\mathbf{x}$ given the observation $\mathbf{y}$. Plugging these expressions into the formula for the posterior of correlated Gaussians in (5.87), we get that

$$\begin{aligned}\mathbf{x}|\mathbf{y} &\sim \mathcal{N}(\mu_x + \Sigma_{xy}\Sigma_y^{-1}(\mathbf{y} - \mu_y), \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}) && (5.136)\\ &\sim \mathcal{N}(\mu_x + \Sigma_x\mathbf{H}^T(\mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z)^{-1}(\mathbf{y} - \mathbf{H}\mu_x), \Sigma_x - \Sigma_x\mathbf{H}^T(\mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z)^{-1}\mathbf{H}\Sigma_x). && (5.137)\end{aligned}$$

So, the MMSE estimate of $\mathbf{x}$ is

$$\hat{\mathbf{x}}_{MMSE} = \mu_x + \Sigma_x\mathbf{H}^T(\mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z)^{-1}(\mathbf{y} - \mathbf{H}\mu_x). \qquad (5.138)$$

The expression for the MMSE estimate is a bit complicated, but we can take apart its components. First, we subtract the mean of $\mathbf{y}$ away from the observation and process it through a "filter" that is quite near the inverse (or pseudo-inverse) of $\mathbf{H}$. Then, we add back the mean $\mu_x$.

The MMSE estimate, as with any Bayes estimator, trades off between the information encoded in the prior and the information present in the observation. A strong prior has a covariance $\Sigma_x$ with small eigenvalues, in which case the filter described by $\Sigma_x\mathbf{H}^T(\mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z)^{-1}$ is small and the MMSE estimate is close to $\mu_x$.

On the other hand, a weak prior has a covariance with large eigenvalues. In this case, the matrix $(\mathbf{H}\Sigma_x\mathbf{H}^T + \Sigma_z)^{-1}$ is approximately $\mathbf{H}\Sigma_x\mathbf{H}^T$. When $\mathbf{H}$ is invertible, the overall filter is therefore approximately $\mathbf{H}^{-1}$; otherwise it is possible to show that the filter converges on the pseudo-inverse $\mathbf{H}^+$. The MMSE estimate trades off between the ML estimate and the prior mean, depending on the relative strength of the prior, the channel, and the noise.

**Example 5.10: White Noise, Isotropic Prior.** *A simple case is one in which the noise covariance is white, i.e. $\Sigma_z = \sigma_z^2\mathbf{I}$, and the prior on $\mathbf{x}$ has zero mean and an isotropic covariance, i.e. $\Sigma_x = \sigma_x^2\mathbf{I}$. This prior supposes that the average energy in each element of the signal $\mathbf{x}$ is $\sigma^2$, but that the elements are* a priori *uncorrelated. In this case, the MMSE estimate is simpler:*

$$\hat{\mathbf{x}}_{MMSE} = \sigma_x^2\mathbf{H}^T(\sigma_x^2\mathbf{H}\mathbf{H}^T + \sigma_z^2\mathbf{I})^{-1}\mathbf{y}. \qquad (5.139)$$

*Here one can clearly see the impact of the prior on the MMSE estimate. A weak prior, with $\sigma_x^2$ large relative to $\sigma_z^2$, converges on the ML estimate $\mathbf{H}^+\mathbf{y}$. The stronger the prior, the more the estimate is "shrunk" towards zero. The prior has the effect of controlling the average power of the solution.*

### 5.3.2 Estimation over Frequency-selective Channels

Even if the channel $\mathbf{H}$ is invertible, it can severely distort the target signal in a way that makes estimation in noise challenging. We will consider the special case, considered in Chapter 4, in which the channel $\mathbf{H}$ corresponds to a **linear time-invariant** system. In this case, recall that the convolution of $\mathbf{x}$ with a filter having impulse response $\mathbf{h} = (h_1, \dots, h_n)$ is denoted by the multiplication with the matrix

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_1 \\ & & \vdots & & \\ h_n & h_1 & h_2 & \cdots & h_{n-1} \end{bmatrix}, \tag{5.140}$$

where we have let $\mathbf{H} \in \mathbb{R}^{n \times n}$, so $\mathbf{y}$ is the noisy ouput of the first $n$ outputs from the filter.[1] We call a matrix with the above structure a **Toeplitz** matrix.

A very useful fact is that a Toeplitz matrix is diagonalized by a so-called **DFT matrix**:

$$\mathbf{H} = \mathbf{U}^H \begin{bmatrix} f_1 & & & \\ & f_2 & & \\ & & \ddots & \\ & & & f_n \end{bmatrix} \mathbf{U}, \tag{5.141}$$

where $\mathbf{U}$ is the matrix corresponding to the discrete Fourier transform:

$$\mathbf{U} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi/n} & e^{-j4\pi/n} & \cdots & e^{-j2\pi(n-1)/n} \\ 1 & e^{-j4\pi/n} & e^{-j8\pi/n} & \cdots & e^{-j4\pi(n-1)/n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j2\pi(n-1)/n} & e^{-j4\pi(n-1)/n} & \cdots & e^{-j2\pi(n-1)^2/n} \end{bmatrix}. \tag{5.142}$$

That is, $\mathbf{U}$ is the matrix such that $\mathbf{Ux}$ gives the discrete Fourier transform of $\mathbf{x}$, and multiplying by $\mathbf{U}^H$ gives the inverse transform. We can look at multiplication by $\mathbf{H}$ as first taking the DFT of $\mathbf{x}$, multiplying it by the frequency response of the LTI system, and then taking the inverse transform. It is not difficult to verify that the DFT matrix is unitary. We can specify an LTI system entirely by the frequency response elements $f_i$, which give the *gain* of the system along the frequency $2\pi i/n$ radians per second.

Suppose that the noise covariance is white, i.e. $\Sigma_z = \sigma_z^2 \mathbf{I}$, and suppose that the system is invertible, meaning that each $f_i > 0$. Then, the maximum likelihood solution is to multiply $\mathbf{y}$ by the inverse of $\mathbf{H}$:

$$\hat{\mathbf{x}}_{ML} = \mathbf{H}^{-1}\mathbf{y} = \mathbf{x} + \mathbf{H}^{-1}\mathbf{z} = \mathbf{x} + \mathbf{U}^H \begin{bmatrix} 1/f_1 & & & \\ & 1/f_2 & & \\ & & \ddots & \\ & & & 1/f_n \end{bmatrix} \mathbf{Uz}. \tag{5.143}$$

That is, each frequency component of the noise is amplified by a factor of $1/f_i$. If the frequency response at any one frequency is small—meaning that the system attenuates those frequencies, then the ML estimator amplifies the noise at these frequencies! This can cause severe estimation errors.

[1] One can elaborate on the signal model to account for filter outputs when the impulse response and signal have

Bayesian estimation allows us to circumvent these problems partially by supposing a prior on the signal. In the simplest case, we can assert a prior with zero mean and a diagonal covariance $\Sigma_x = \sigma_x^2 \mathbf{I}$ that dictates the average energy of the signal. Then, the MMSE estimate is

$$\begin{aligned}
\hat{\mathbf{x}}_{MMSE} &= \sigma_x^2 \mathbf{H}^T (\sigma_x^2 \mathbf{H}\mathbf{H}^T + \sigma_z^2 \mathbf{I})^{-1} \mathbf{y} && (5.144)\\
&= \sigma_x^2 \mathbf{U}^H \mathbf{F}^* \mathbf{U} (\mathbf{U}^H (\sigma_x^2 |\mathbf{F}|^2 + \sigma_z^2 \mathbf{I}) \mathbf{U})^{-1} \mathbf{y} && (5.145)\\
&= \sigma_x^2 \mathbf{U}^H \mathbf{F}^* (\sigma_x^2 |\mathbf{F}|^2 + \sigma_z^2 \mathbf{I})^{-1} \mathbf{U}\mathbf{y} && (5.146)\\
&= \mathbf{U}^H \left( \mathbf{F} + \frac{\sigma_z^2}{\sigma_x^2} \mathbf{F}^* \right)^{-1} \mathbf{U}\mathbf{y}. && (5.147)
\end{aligned}$$

The resulting estimator is similar to the ML estimator. In principle, one takes the Fourier transform of the observation, and "almost" inverts the frequency response of the channel before taking the inverse transform. The term $\frac{\sigma_z^2}{\sigma_x^2}$ ensures that the gain at any one frequency does not become too large, according to the signal-to-noise ratio indicated. For small $\sigma_z^2$ or large $\sigma_x^2$, the estimator is approximately equal to the ML estimator.

### 5.3.3 Using Frequency-dependent Priors

In a similar manner, one can choose a prior distribution that is frequency-dependent, by letting $\mu_x = 0$ and

$$\Sigma_x = \mathbf{U}^H \begin{bmatrix} f_1 & & & \\ & f_2 & & \\ & & \ddots & \\ & & & f_n \end{bmatrix} \mathbf{U}, \tag{5.148}$$

where again $\mathbf{U}$ is the DFT matrix. The choice of $f_i$s allows one to localize the signal in frequency. A common choice is to suppose a **low-pass** model on signals, in which the $f_i$s corresponding to low frequencies are higher than the $f_i$s corresponding to low frequencies, which may even be set to zero.

In this case, performing MMSE estimation in effect applies a low-pass (or otherwise frequency-selective) filter to the output of the channel inverse. Note that this prior can be used even if $\mathbf{H}$ is not Toeplitz.

### 5.3.4 Using Data-driven Priors

Finally, we can choose our prior based on previous datasets that we believe are representative of future data. For example, given a dataset of natural images, we can estimate the sample mean and covariance and choose them as $\mu_x$ and $\Sigma_x$. When applied to noisy measurements, such a prior will result in an estimator that prefers to generate images that look like those in the dataset.

Furthermore, one can choose a "low-pass" prior by taking only the dominant principal components as the prior covariance. In this case, the estimation is towards low-pass signals over a data-driven basis set, rather than over the Fourier basis as described above.

different lengths, or to describe the filter output past the $n$th sample. We will ignore these issues for simplicity.

### 5.3.5 Sparse Signal Processing

In this section, we'll look at Bayesian estimation under a **sparse** prior, meaning that the signal is supposed to be a linear combination of just a few elements in a basis set. This is rather similar to PCA, but with important differences that we'll see shortly. Sparse signal processing and learning is one of the great breakthroughs in the past decade or two.

#### 5.3.5.1 Priors as Regularization

Before we get to defining formally a sparse signal and a sparse prior, let's dive a bit deeper into the impact of the prior on estimation. To make things concrete, let's suppose that we have the linear measurement model

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}, \tag{5.149}$$

where the noise covariance is supposed to be white, i.e. $\Sigma_z = \sigma_z^2\mathbf{I}$. Then, the likelihood function for $\mathbf{y}$ is $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$. Finding the maximum-likelihood estimate therefore corresponds to the optimization problem:

$$\begin{aligned}
\hat{\mathbf{x}}_{ML} &= \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) && (5.150)\\
&= \arg\max_{\mathbf{x}} \log(p(\mathbf{y}|\mathbf{x})) && (5.151)\\
&= \arg\max_{\mathbf{x}} -\frac{1}{2\sigma_z^2}||\mathbf{y} - \mathbf{H}\mathbf{x}||^2 && (5.152)\\
&= \arg\min_{\mathbf{x}} ||\mathbf{y} - \mathbf{H}\mathbf{x}||^2. && (5.153)
\end{aligned}$$

In other words, the ML estimator finds the $\mathbf{x}$ that minimizes the Euclidean distance between $\mathbf{y}$ and $\mathbf{H}\mathbf{x}$. We know that this least squares solution is found by applying the (pseudo)-inverse of $\mathbf{H}$ to $\mathbf{x}$.

Now, let's assert a prior on $\mathbf{x}$ of $p(\mathbf{x}) = \mathcal{N}(0, \sigma_x^2\mathbf{I})$. By conjugacy, the joint and posterior distributions are Gaussian. The MAP and MMSE estimates are the same. Rather than solving directly for the estimator as we did previously, let's think about the optimization problem corresponding to findin the MAP (and therefore MMSE) estimator:

$$\begin{aligned}
\hat{\mathbf{x}}_{MMSE} &= \arg\max_{\mathbf{x}} \log(p(\mathbf{y}|\mathbf{x})) + \log(p(\mathbf{x}))\\
&= \arg\max_{\mathbf{x}} -\frac{1}{2\sigma_z^2}||\mathbf{y} - \mathbf{H}\mathbf{x}||^2 - \frac{1}{2\sigma_x^2}||\mathbf{x}||^2\\
&= \arg\min_{\mathbf{x}} \frac{1}{2\sigma_z^2}||\mathbf{y} - \mathbf{H}\mathbf{x}||^2 + \frac{1}{2\sigma_x^2}||\mathbf{x}||^2\\
&= \arg\min_{\mathbf{x}} ||\mathbf{y} - \mathbf{H}\mathbf{x}||^2 + \frac{\sigma_z^2}{\sigma_x^2}||\mathbf{x}||^2.
\end{aligned}$$

In finding the MMSE estimator, we try simultaneously to minimize the Euclidean distance between $\mathbf{y}$ and $\mathbf{H}\mathbf{x}$ and the norm of $\mathbf{x}$. Another way to put this is that we **regularize** the minimization of the squared error by the norm of $||\mathbf{x}||^2$. This regularization trades off between minimizing $||\mathbf{y} - \mathbf{H}\mathbf{x}||^2$ and minimizing $||\mathbf{x}||^2$, with the trade-off decided by the strength of the prior relative to the noise variance. In optimization, we often call $||\mathbf{x}||^2$ the **regularizing function** or **penalty function**, and the **Lagrange multiplier** $\frac{\sigma_z^2}{2\sigma_x^2}$ determines how strong one penalizes solutions with large norms.

In this sense, the prior **regularizes** the estimation problem. In estimation and signal processing it is common to impose a regularizing function to make sure that the solution is not pathological. Often these regularizers are imposed directly, without any sense of a prior or likelihood function. In fact, the MMSE estimator above is often called **Tikhonov regularization**, a common variant of least-squares estimation that does not explicitly assume a statistical model for the data. In a moment we will see how other regularizers emerge from the use of non-Gaussian priors on $\mathbf{x}$.

#### 5.3.5.2 Sparse Signal Models

We introduce the concept of sparsity with the following definition.

**Definition 5.3:.** *We say that the signal* $\mathbf{x} \in \mathbb{R}^m$ *is* **k-sparse** *with respect to the* **dictionary** $\mathbf{W} \in \mathbb{R}^{m \times l}$ *if there is a vector* $\mathbf{c} \in \mathbb{R}^l$ *such that*

$$\mathbf{x} = \mathbf{W}\mathbf{x} \tag{5.154}$$

$$||\mathbf{c}||_0 \leq k, \tag{5.155}$$

*where* $||\mathbf{c}||_0$ *is the* $\ell_0$ **norm** *of* $\mathbf{c}$*, which just counts the number of non-zero entries in* $\mathbf{c}$.

That is, a signal $\mathbf{x}$ is k-sparse with respect to the dictionary $\mathbf{W}$ provided $\mathbf{x}$ is a linear combination of no more than $k$ columns of $\mathbf{W}$. We will often refer to the columns of $\mathbf{W}$ as the **atoms** in the dictionary. We also will often refer to $\mathbf{W}$ as the **sparsifying dictionary**, the **sparsifying basis**, or occasionally just the **basis**.

One might wonder why were are concerned with sparsity at all—can we really expect signals to be a linear combination of just a few elements? It turns out empirically that we often can. For example, if $\mathbf{W}$ is the Fourier basis, then a sparse audio signal is one that is composed of just a few tones, which is a reasonable model for many practical signals. During the 1980s and 1990s, signal processing researchers found that audio and image signals are often sparse in bases such as the **discrete cosine transform** (DCT) and **discrete wavelet transform** (DWT). Researchers in the 1990s also experimented with custom-defined bases, such as "curvelets" and "edgelets", that allow for a sparse representation of more complicated image features. That signals can be well approximated in such bases is the fundamental idea behind lossy compression schemes such as JPEG, MPEG, etc.

We will define a few important dictionaries in a moment, but first we pause to distinguish two important types. The first is a **complete dictionary**. When $l = m$ and $\mathbf{W}$ has full rank, then we say that $\mathbf{W}$ is **complete**. We will virtually always assume that the columns of $\mathbf{W}$ are orthonormal in this case, in which case $\mathbf{W}$ is an orthonormal basis for $\mathbb{R}^m$. A few examples are:

- The **discrete Fourier transform** (DFT) basis, which is the set of discrete-time complex exponential used in computing the DFT. We saw the DFT basis earlier in this section.
- The **discrete cosine transform** (DCT) basis, which is a two-dimensional basis similar to the DFT basis and is useful for images. The DCT basis is used in JPEG compression of images.
- The **discrete wavelet transform** (DWT) basis. The DWT is based on integral transforms that use functions called **wavelets** in place of sinusoids or complex exponentials. Wavelets have a large and rich body of literature, which we do not have time to explore here. But

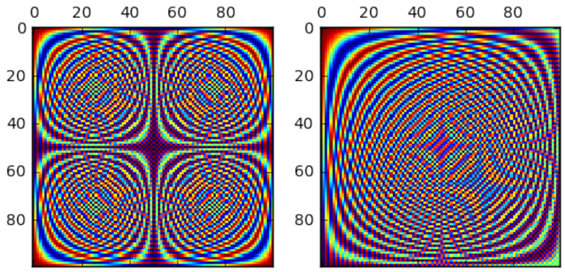

Figure 5.3: DFT and DCT bases for length-100 signals. For the DFT basis, only the real part is shown.

the upshot is that the DWT provides a *multi-scale* representation of a signal, looking at it at different timescales simultaneously. The DWT basis is used in the JPEG-2000 standard.

Examples of the first two of these bases are shown in Figure **??**.

The second type is an **overcomplete dictionary**. When $l > m$, we say that $\mathbf{W}$ is **overcomplete**, meaning that we can delete atoms from $\mathbf{W}$ and still have a complete dictionary. It is intuitively obvious that an overcomplete basis is more likely to sparsify a set of signals; the more atoms there are, the more likely we are to be able to represent a signal with only a few of them. A variety of "designed" overcomplete dictionaries have been proposed. These include concatenations of DCT and DWT dictionaries, as well as "augmenting" a complete dictionary by taking transformations of existing atoms. A common example in the literature is the so-called "overcomplete" DCT basis, in which the DCT atoms are combined to form new atoms. A more modern approach is to learn an overcomplete dictionary directly from the data. There are a few successful methods for this, including the **k-SVD** algorithm, which we will look at in a bit more detail later.

At this point it's worthwhile to point out the differences between sparse coding and PCA. In both cases, we model a signal as a linear combinations of only a few components from a basis set. However, in PCA, we suppose that signals are linear combinations of the *same* few elements. In sparse coding, however, each signal may be a linear combination of a different set of atoms. We can think of PCA as coding over an **undercomplete** basis.

Now that we have a handle on sparse signal representations, we need to find a way to model them probabilistically. In particular, we want to suppose a *prior* over the signal that will induce sparsity with respect to a known dictionary $\mathbf{W}$. A perhaps surprising fact is that the Laplace prior induces sparsity:

$$p(\mathbf{c}) = \frac{\lambda}{2} \exp\left(-\lambda\left(\sum_{i=1}^{l} |c_i|\right)\right). \tag{5.156}$$

In Figure 5.4, we plot i.i.d. samples from the Laplace distribution. Note that none of the samples are exactly zero; this signal is only approximately sparse. However, we will see in a moment that an approximately sparse prior is enough to carry out sparse signal processing.

#### 5.3.5.3 Sparse Signal Coding: The LASSO

Given a signal $\mathbf{x} \in \mathbb{R}^m$ that we believe is sparse with respect to $\mathbf{W}$, how do we find the sparse coefficients $\mathbf{c}$? Let's suppose that $\mathbf{x}$ is sparse up to white Gaussian noise:

$$\mathbf{x} = \mathbf{W}\mathbf{c} + \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}). \tag{5.157}$$

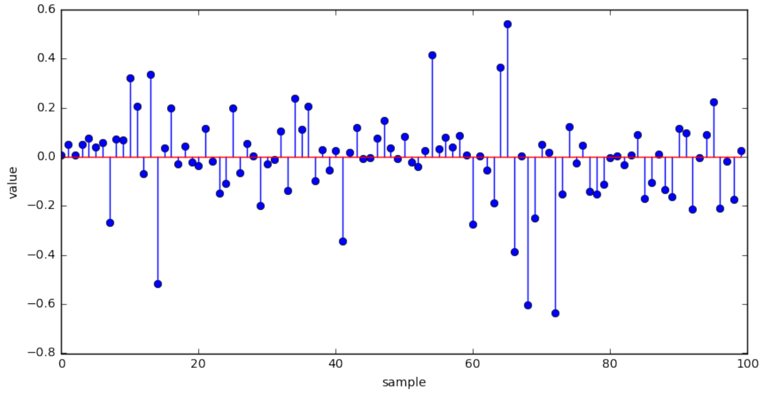

Figure 5.4: 100 samples from the Laplace distribution, in this case with $\lambda = 10$.

Thus, we have the likelihood function $p(\mathbf{x}|\mathbf{c}) = \mathcal{N}(\mathbf{Wc}, \mathbf{I})$. If we suppose a Laplace prior on $\mathbf{c}$, we can pose the MAP (but not MMSE!) estimate of $\mathbf{x}$ as the following optimization problem:

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \log(p(\mathbf{x}|\mathbf{c})) + \log(p(\mathbf{c})) \tag{5.158}$$

$$= \arg\max_{\mathbf{c}} -||\mathbf{x} - \mathbf{Hc}||^2 - \frac{\lambda}{2}\sum_{i=1}^{l} |c_i| \tag{5.159}$$

$$= \arg\min_{\mathbf{c}} ||\mathbf{x} - \mathbf{Hc}||^2 + \frac{\lambda}{2}||\mathbf{c}||_1, \tag{5.160}$$

where $||\mathbf{c}||_1 = \sum_i |c_i|$ is called the $\ell_1$ **norm** of $\mathbf{c}$.

The optimization problem posed above is an extremely important formulation called the **LASSO**, which stands for *least absolute shrinkage and selection operator*. LASSO was originally formulated in a slightly different form in the context of sparse linear regression—indeed, we'll revisit the LASSO in this context later—and it is virtually impossible to understate its importance and impact on statistics, signal processing, and machine learning. In signal processing the problem is often called **basis pursuit denoising**, where the formulation is virtually exactly as we stated above. In fact, BDPN was introduced first—just two years before the LASSO!—but the problems are so near to identical that many authors will use the two terms interchangeably. We will dive a bit deeper into this question later.

However, we merely posed an optimization problem, and haven't given its solution! It is straightforward to verify that the objective function in the above is convex, so we are guaranteed to be able to find a global solution. However, unlike the previous cases, we cannot solve in closed form for $\hat{\mathbf{x}}$. In fact, we cannot directly use gradient methods, because the derivative of $|x|$ does not exist at $x = 0$. Standard convex optimization libraries, as available with MATLAB, can solve the problem quickly. There are a number of specially-designed algorithms that solve LASSO/BPDN as well, including those built into scikit-learn. We will not go into the details of these solvers; for our purposes it is enough to see that this problem is convex and therefore has a solution.

Sparse coding is another example of estimating with a regularizer. We look for the coefficient vector $\mathbf{c}$ that best describes the signal $\mathbf{x}$, but since we are after a sparse solution, we penalize solutions with large $\ell_1$ norm. This leads to solutions that simultaneously are descriptive of $\mathbf{x}$ and sparse. The sparsity of the solution is controlled by the strength of the prior; the larger $\lambda$, the sparser the solution.

A natural question is whether or not we could have solved this problem with a penality on the $\ell_0$ norm:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} ||\mathbf{x} - \mathbf{H}\mathbf{c}||^2 + \frac{\lambda}{2}||\mathbf{c}||_0. \tag{5.161}$$

After all, if $\mathbf{x}$ is sparse, why not look directly for a sparse solution by penalizing solutions with many non-zero coefficients? It turns out that solving this problem is not only non-convex, but also NP hard, and there is no efficient way to find a solution in general. Indeed, the LASSO/BDPN can be thought of as a *convex relaxtion* of the above problem: a convex problem that is near enough that its solution is useful. It turns out that there is a nice body of work showing that for "almost" every sparse vector, $\ell_1$ minimization exactly recovers it.

#### 5.3.5.4 Sparse Signal Recovery

Let's revisit the noisy signal recovery problem, but this time let's suppose the signal is sparse with respect to a known dictionary, i.e. $\mathbf{x} = \mathbf{W}\mathbf{c}$ for $\mathbf{c}$ sparse. Then, the signal $\mathbf{x}$ is completely specified by the vector of coefficients $\mathbf{c}$. In this case, the signal recovery problem reduces to the problem of finding the sparse coefficients.

As before, we suppose the the signal is passed through a system $\mathbf{H}$ and corrupted by Gaussian noise:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z} = \mathbf{H}\mathbf{W}\mathbf{c} + \mathbf{z}, \tag{5.162}$$

so the likelihood function is $p(\mathbf{y}|\mathbf{c}) = \mathcal{N}(\mathbf{H}\mathbf{W}\mathbf{c}, \mathbf{I})$. If we assume a Laplace prior on $\mathbf{c}$, we can express the MAP estimate of $\mathbf{c}$:

$$\begin{aligned}\hat{\mathbf{c}} &= \arg\max_{\mathbf{c}} \log(p(\mathbf{y}|\mathbf{c})) + \log(p(\mathbf{c}))\\ &= \arg\min_{\mathbf{c}} ||\mathbf{y} - \mathbf{H}\mathbf{W}\mathbf{c}||^2 + \frac{\lambda}{2}||\mathbf{c}||_1.\end{aligned}$$

Again the solution has the form of the LASSO/BPDN, and again this problem can be solved effectively via numerical methods.

Note that this is similar to the sparse coding case, except now the equivalent dictionary is $\mathbf{H}\mathbf{W} \in \mathbb{R}^{n\times l}$. Whether or not we can recover $\mathbf{c}$ (and therefore $\mathbf{x}$) from $\mathbf{y}$ depends on whether or not the equivalent dictionary $\mathbf{H}\mathbf{W}$ is sufficiently well behaved. A landmark result in signal processing is that, under certain conditions on the structure of $\mathbf{H}\mathbf{W}$, recovery of $\mathbf{x}$ is possible as long as the number of measurements $n$ is on the order of $k\log(l)$. This remarkable result is the basis of **compressive sensing**.

#### 5.3.5.5 Dictionary Learning: k-SVD

In PCA, we learned a(n undercomplete) basis that described a group of signals by forming the empirical covariance matrix and taking the dominant eigenvectors. In this section, our objective is to learn an overcomplete, sparsifying basis for a large set of data.

Suppose we have access to $s$ data samples $\mathbf{x}_i \in \mathbb{R}^m$, and our objective is to learn a dictionary $\mathbf{W} \in \mathbb{R}^{m\times l}$ that sparsifies the data samples. That is, we want to find both a dictionary $\mathbf{W}$ such that for every $\mathbf{x}_i$ there is a sparse coefficient $\mathbf{c}_i$ such that $\mathbf{W}\mathbf{c}_i \approx \mathbf{x}_i$. We can pose this problem as a regularized optimization problem:

$$(\mathbf{W}^*, \mathbf{c}^*) = \arg\min_{\mathbf{W},\mathbf{c}} \sum_{i=1}^{s} ||\mathbf{x}_i - \mathbf{W}\mathbf{c}_i||^2 + \lambda \sum_{i=1}^{s} ||\mathbf{c}_i||_1. \tag{5.163}$$

That is, we solve simultaneously for sparse coefficient vectors that code each sample and a dictionary that admits a sparse coding of each sample.

It turns out that this optimization problem is non-convex. This may be a bit surprising, since finding the optimum sparse coefficient vectors given a fixed $\mathbf{W}$ is convex. However, jointly finding $\mathbf{W}$ and the coefficient vectors $\mathbf{c}_i$ is non-convex. As a result, a variety of iterative algorithms have been proposed to find a local optimum. Perhaps the most popular of these is **k-SVD**. It operates by alternating minimization similar to k-means clustering: Given a fixed dictionary, the optimum $\mathbf{c}_i$s are founding using (for example) BPDN; then, given fixed $\mathbf{c}_i$s, the dictionary $\mathbf{D}$ is updated to give a better approximation of the data samples. As with LASSO, we will not dive into the details of this algorithm.

Although k-SVD is not guaranteed to converge to a global optimum, in practice it works out well. There is an implementation of an algorithm similar to k-SVD in scikit-learn.

## 5.4 Bayesian Regression

In this section we will look at Bayesian approaches to regression. In each case, this will amount to supposing a prior and letting that prior regularize the optimization problem associated with the regression problem. We will focus on two cases: **ridge regression**, where we suppose a Gaussian prior on the regression coefficients and regularize their square, and **LASSO**, where we suppose a Laplace prior the regression coefficients and seek a sparse solution. The resulting approaches are similar in spirit to the MMSE/Tikhonov and $\ell_1$/BPDN estimators from the previous section.

Before proceeding, it is worth asking why to bother with elaborations on linear regression at all. After all, the solution we derived above is the MVUE if the data are truly Gaussian, and the best linear unbiased estimator otherwise. Isn't this enough? However, researchers have long noted issues with regression, which are addressed by ridge regression and the LASSO. We'll focus on the following:

- **Interpretability.** When training a large regression model, we don't just want to know how to map the inputs to a good prediction of the outputs. We (may) also want to be able to interpret the regression. In, for example, regression over medical or genetic data, we want to know which risk factors make the biggest impact on outcomes. In this case, it is more useful to learn a model that has a few large regression coefficients.

- **Small Sample Size.** When the number of data samples is smaller than the number of input factors, linear regression is ill-posed. We need to regularize the problem in order to get a solution. However, even if the number of samples is strictly large enough for a valid solution, the variance of the estimate may be high, leading to poor predictive performance. In each case, a prior can be introduced to decrease the variance of the solution and to ensure a well-posed problem.

### 5.4.1 Ridge Regression

We will start with ridge regression. For simplicity we consider the standard *scalar* regression scenario. We have $n$ samples of the input $\mathbf{x}_i \in \mathbb{R}^d$, and we let $\mathbf{y} \in \mathbb{R}^n$ be the output or response vector. Rather than define a loss function directly, we will assume a probabilistic model relating the input and the output:

$$y = \mathbf{b}^T\mathbf{x} + z, \tag{5.164}$$

where $z \sim \mathcal{N}(0, \sigma^2)$ and $\mathbf{b} \in \mathbb{R}^d$ is the vector of regression coefficients. Therefore, the likelihood function is $p(y|\mathbf{x}, \mathbf{b}) = \mathcal{N}(\mathbf{b}^T\mathbf{x}, \sigma^2)$. We can aggregate all $n$ input samples into a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, in which case we get the relationship

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{z}, \tag{5.165}$$

where now $\mathbf{z} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, and the resulting likelihood function is $p(\mathbf{y}|\mathbf{X}, \mathbf{b}) = \mathcal{N}(\mathbf{X}\mathbf{b}, \sigma^2\mathbf{I})$. We can find the standard least-squares estimate of the regression coefficients as long as $n \leq d$:

$$\hat{\mathbf{b}} = \mathbf{X}^+\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5.166}$$

In ridge regression, we suppose a Gaussian prior on $\mathbf{b}$:

$$\mathbf{b} \sim \mathcal{N}(0, \tau), \tag{5.167}$$

for some $\tau > 0$. Then, the posterior for $\mathbf{b}$ can be found using Bayes rule:

$$p(\mathbf{b}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{b})p(\mathbf{X})p(\mathbf{b})}{p(\mathbf{X}, \mathbf{y})} \propto p(\mathbf{y}|\mathbf{X}, \mathbf{b})p(\mathbf{b}). \tag{5.168}$$

We can find the MMSE/MAP solution by optimizing over the posterior:

$$\begin{aligned}\mathbf{b}^* &= \arg\max_{\mathbf{b}} -\frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 - \frac{1}{2\tau}||\mathbf{b}||^2 &\quad (5.169)\\ &= \arg\min_{\mathbf{b}} ||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 + \frac{\sigma^2}{\tau}||\mathbf{b}||^2. &\quad (5.170)\end{aligned}$$

This optimization problem is similar to Tikhonov regularization seen in the previous chapter. We seek the $\mathbf{b}$ that minimizes the squared error, penalized by the norm of the regression coefficients. This “shrinks” the coefficients towards the origin, encouraging smaller coefficients, especially if there are a few large coefficients that can explain most of the data. It is straightforward in this case to set the gradient equal to zero and find the closed-form solution:

$$\mathbf{b}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\sigma^2}{\tau}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{5.171}$$

The identity matrix we introduce regularizes the solution, making it less dependent on the data. For starters, this means that we can solve the regression problem even when $n < d$. Furthermore, if $\tau$ is large relative to $\sigma^2$, the resulting estimator matrix has small eigenvalues, which shrinks the solution towards the origin.

Ridge regression also can be posed without resorting to a Bayesian model. One can directly pose a loss function that incorporates both squared error and the norm of the coefficients:

$$l(f(\mathbf{x}), y) = (y - \mathbf{b}^T\mathbf{x})^2 + \lambda||\mathbf{b}||^2. \tag{5.172}$$

In this case, the empirical risk over $n$ labeled samples is

$$L_n(f) = ||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 + \lambda||\mathbf{b}||^2, \tag{5.173}$$

and the ERM solution is exactly the MMSE solution, slightly reparameterized:

$$\mathbf{b}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5.174}$$

The benefit of framing the problem this way is that it makes no assumptions about the data distribution.

### 5.4.2 LASSO

We introduced the LASSO previously as a moniker for sparse coding of signals. As we noted then, LASSO was originally proposed as a means for regularized regression. The setup and solution is similar in spirit to ridge regression, except that instead of looking for a solution with small norm, we are looking for one that is sparse. A sparse solution to regression is particularly useful for constructing an interpretable model. If there are only a few non-zero elements of $\mathbf{b}$, then we have identified the few critically important factors in predicting the response.

To frame the problem, as before we suppose a model in which $n$ input samples $\mathbf{x}_i \in \mathbb{R}^d$ and response scalars $y_i$ are supposed to be related via a noisy linear model:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{z}, \tag{5.175}$$

where again the resulting likelihood function is $p(\mathbf{y}|\mathbf{X}, \mathbf{b}) = \mathcal{N}(\mathbf{X}\mathbf{b}, \sigma^2\mathbf{I})$. We suppose a Laplace prior on the coefficients $\mathbf{b}$, and similar to before we end up with a regularized least squares MAP estimate:

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \log(p(\mathbf{y}|\mathbf{X}, \mathbf{b})) + \log(p(\mathbf{b})) \tag{5.176}$$

$$= \arg\min_{\mathbf{b}} ||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 + \frac{\sigma^2}{\lambda}||\mathbf{b}||_1. \tag{5.177}$$

Note that this is *not* the MMSE estimate, since the posterior is not Gaussian. We can reparameterize the solution to simplify the penalty term:

$$\mathbf{b}^* = \arg\min_{\mathbf{b}} ||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 + \lambda||\mathbf{b}||_1. \tag{5.178}$$

As we saw in BPDN, this problem is convex but does not have a closed-form solution. A variety of efficient solvers exist for finding the solution to the LASSO. The important upshot is that one can use sparsity to "cut through" irrelevant data features that do not contribute much to the output. The results is an easily interpreted model. Furthermore, LASSO can be used when the number of data samples is relatively small; sparsity means that we need to see only a few examples to learn the regression coefficients.

Similarly, one can define the loss function that leads to LASSO without supposing a statistical data model. Letting

$$l(f(\mathbf{x}), y) = (y - \mathbf{b}^T\mathbf{x})^2 + \lambda||\mathbf{b}||_1, \tag{5.179}$$

the empirical risk over $n$ labeled samples is

$$L_n(f) = ||\mathbf{y} - \mathbf{X}\mathbf{b}||^2 + \lambda||\mathbf{b}||_1, \tag{5.180}$$

and the ERM solution is exactly the LASSO solution.

# Chapter 6

# The Kalman Filter

In the previous chapter we studied the estimation of a *static* parameter: There is some (perhaps vector) parameter $\theta$, which we estimate indirectly via the observation $Y$. In this chapter, we study the estimation of a *dynamic* process. Examples of dynamic processes include the trajectory (position, velocity, etc.) of a moving body over time, or the evolution of the impulse response of a linear (but time-variant) system. We will formalize the notion of a dynamic process by the discrete-time, **linear state space** model: The state of the process is modeled by a vector that varies in discrete time, and the state vector at time $t + 1$ is a noisy linear combination of the state vector at time $t$. Furthermore, we will suppose that, rather than observing the state vector directly, we will obtain a measurement that is a noisy linear combination of the state vector. We will focus on the case in which the noise on the dynamic model and the observation noise is Gaussian. A suprising number of realistic systems fall under the linear/Gaussian state space model.

How can we best estimate the dynamic state vector? One one hand, we could use the linear observation model to estimate the state vector at each time step individually. At each time step $t$, we could use the techniques from the previous chapter to construct an MMSE or ML estimate of the state vector given the noisy observation. However, estimating each state vector individually throws out vital information: the state vector evolves in time according to a known linear dynamic model! If we have a good estimate of the state vector at time $t$, we ought to be able to predict the state vector at time $t + 1$. How can we incorporate this information into the estimation process?

The **Kalman filter** is the answer to this question. It provides the optimum Bayesian estimator of a linear/Gaussian dynamic process in all of the usual senses: It minimizes the mean-squared error, satisfies the maximum *a priori* condition, etc. We will derive the Kalman filter in this chapter, showing that a two-step process provides the Bayes-optimum estimate of the state vector. First, we compute the Bayes-optimum prediction of the state vector at time $t$ given our estimate of the state at time $t - 1$. Then, we update this estimate by computing the Bayes-optimum estimate of the state vector given the noisy observation we obtained at time $t$. Each of these steps uses the results on vector Gaussian estimation we derived in the previous chapter.

The result is an estimator that finds the optimal trade-off between what our observations and the dynamic state space model tell us about the state we wish to track. The Kalman filter is ubiquious in engineering, appearing in applications as mundane the cellphone and as earth-shattering as the Apollo program.

## 6.1 Linear State Space Models

### 6.1.1 Browninan Motion

Before we introduce state space models in their full generality, let's consider a simple example that you have likely seen before: **Brownian motion**. Brownian motion has origins going back to Einstein, who applied the model to the motion of particles suspended in fluid. We'll look at the scalar case of Brownian motion, in which $x_t \in \mathbb{R}$ models the position of a particle in one-dimensional space at time $t$. We suppose that the motion of the particle between time $t$ and $t+1$ is modeled by Gaussian noise:

$$x_t = x_{t-1} + z_t, \tag{6.1}$$

where $z_t \sim \mathcal{N}(0, \sigma^2)$. We want to track the sequence $x_0, x_2, \cdots$ of particle positions over time. Suppose we know the initial conditions, and even more specifically let's say that $x_0 = 0$ is known to us. If the noise variance $\sigma^2$ is small, we can guess that $x_1 \approx x_0$ without too much error. However, we're stuck guessing that $x_2$ is close to $x_1$, thus $x_2 \approx 0$, and so on. As more time passes, the errors will begin to accumulate, and our estimate will get further and further from the true position.

So, let's suppose we have a device that measures the position of the particle. This measurement is imperfect and therefore subject to noise. We model this as follows:

$$y_t = x_t + n_t, \tag{6.2}$$

where $n_t \sim \mathcal{N}(0, \sigma_n^2)$ is the measurement noise. Again, if the measurement noise $\sigma_n^2$ is small, we can guess $x_t = y_t$. However, this ignores the fact that the state at time $x_t$ is related to the state at time $x_{t-1}$! If the observation gave us a good estimate of $x_{t-1}$, and if the noise variance $\sigma^2$ is small, we ought to be able to incorporate that fact into our estimate.

One way to think of this is that the estimation problem has two components. Because the noise on Brownian motion is Gaussian, we have

$$p(x_t | x_{t-1}) = \mathcal{N}(x_{t-1}, \sigma^2). \tag{6.3}$$

This gives us the posterior distribution of $x_t$ given $x_{t-1}$, from which one could compute the MMSE estimator. Again, the MMSE estimate is boring—it's just $x_t$!—so intuitively we know the MMSE estimate isn't sufficient. On the other hand, the measurement noise is Gaussian, so

$$p(y_t | x_t) = \mathcal{N}(x_t, \sigma_n^2). \tag{6.4}$$

This distribution is more familiar—it's the likelihood function for the observation given the state we want to estimate. Using techniques from the previous chapter, we can construct the Bayes or MVUE estimator of $x_t$ given $y_t$. (This estimate is boring, too: $y_t$!)

The challenge of the sequential estimation problem, which we will solve when we derive the Kalman filter, is how to combine these separate estimation problems—which we know how to solve individually—into a joint solution.

### 6.1.2 A Kinematic Example

Let's look at one more simple example to motivate the state-space model. Similar to the case of Brownian motion, we are interested in tracking the position of an object, although here we consider an object in three-dimensional space. Furthermore, we'll consider the effect of the dynamics of the object position: if we know the object's *velocity* in addition to its position at time $t$, we can predict accurately its position at time $t+1$. We model this by defining a state vector with six elements, three for the object position and three for the velocity:

$$x_t = (p_t^x, p_t^y, p_t^z, v_t^x, v_t^y, v_t^z). \tag{6.5}$$

We need to be careful with kinematic models in discrete time. Let's suppose that the discrete time steps are $\Delta$ seconds apart, and let's also suppose that the velocity is constant over the interval between time steps. Finally, we'll assume that the object is subject to random accelerations—forces due to friction, etc. that we cannot model deterministically. We can model this with the following equations:

$$\begin{bmatrix} p_{t+1}^x \\ p_{t+1}^y \\ p_{t+1}^z \end{bmatrix} = \begin{bmatrix} p_t^x \\ p_t^y \\ p_t^z \end{bmatrix} + \Delta \begin{bmatrix} v_t^x \\ v_t^y \\ v_t^z \end{bmatrix}, \quad \begin{bmatrix} v_{t+1}^x \\ v_{t+1}^y \\ v_{t+1}^z \end{bmatrix} = \begin{bmatrix} v_t^x \\ v_t^y \\ v_t^z \end{bmatrix} + \begin{bmatrix} z_t^x \\ z_t^y \\ z_t^z \end{bmatrix}, \tag{6.6}$$

where each $z_t^i \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise that models the random accelerations. In other words, at each time step the object position moves in proportion to its velocity, and the velocity remains constant except for random perturbations.

Getting closer to general linear state space models, we can write the dynamics in a single matrix-vector equation:

$$\begin{bmatrix} p_{t+1}^x \\ p_{t+1}^y \\ p_{t+1}^z \\ v_{t+1}^x \\ v_{t+1}^y \\ v_{t+1}^z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t+1}^x \\ p_{t+1}^y \\ p_{t+1}^z \\ v_{t+1}^x \\ v_{t+1}^y \\ v_{t+1}^z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ z_t^x \\ z_t^y \\ z_t^z \end{bmatrix}. \tag{6.7}$$

Just like in the Brownian motion example, let's suppose we get to measure the position—but not the velocity—of the object, subject to noise. We write this with the following equation:

$$y_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{t+1}^x \\ p_{t+1}^y \\ p_{t+1}^z \\ v_{t+1}^x \\ v_{t+1}^y \\ v_{t+1}^z \end{bmatrix} + \begin{bmatrix} n_t^x \\ n_t^y \\ n_t^z \end{bmatrix}, \tag{6.8}$$

where each $n_t^i \sim \mathcal{N}(0, \sigma_n^2)$ is additive Gaussian noise.

As in the case of Brownian motion, we have two easy estimation problems that we want to combine in an optimal fashion. If we know the entire state vector at time $t$, we can make a good prediction of the state at time $t+1$ by propagating the velocity by one time step. On the other hand, the measurement $y_t$ gives us a good estimate of the position, but not the velocity. A nice feature of the Kalman filter is that it will allow us to combine the dynamics of the moving object with measurements of the object position to estimate both the position and velocity of the object, even though we measure only the position directly. The velocity estimate, albeit indirect, allows us to keep track of the object position more accurately than possible if we ignore the velocity altogether.

### 6.1.3 General State Space Models

More generally, we can define the linear Gaussian state space model. Let the state vector $x_t \in \mathbb{R}^n$ be $n$-dimensional, and let the observation $y_t \in \mathbb{R}^d$ be $d$-dimensional. Then, we model the evolution of the state and the observation by the following two equations:

$$x_{t+1} = F_t x_t + G_t u_t \tag{6.9}$$

$$y_t = H_t x_t + v_t. \tag{6.10}$$

Let's take a moment to unpack the different components of this model. The matrix $F_t \in \mathbb{R}^{n\times n}$ is the linear system that describes the evolution of the state vector. In Brownian motion, $F_t = 1$, and in the kinematic model the matrix $F_t$ was more complicated. A general linear model may account for a variety of factors: a more complicated kinematics may incorporate acceleration, or forces like gravity; they may also represent dynamics other than motion through space, such as the evolution of the channel describing the propagation of a wireless signal. The matrix $H_t$ describes the linear measurement model: in simple models like those we've looked at, the observation is simply the position of the object corrupted by noise; other possible models include the details of an optical system for imaging.

The random vector $u_t \in \mathbb{R}^s$ is an $s$-dimensional Gaussian vector with zero mean and covariance matrix $Q_t$. This random vector, multiplied by the matrix $G_t \in \mathbb{R}^{n\times s}$, describes the random component of the evolution of the state vector. In Brownian motion $s = 1$, the matrix $G_t = 1$ is a scalar; in the kinematic model $s = 3$ and the matrix $G_t$ maps the random accelerations to changes in the velocity components of the state vector. Finally, the random vector $v_t \in \mathbb{R}^d$ is Gaussian with zero mean and covariance matrix $R_t$, and it accounts for noise in the observation model.

So far, we have considered state space models that are *time invariant*, meaning that $F_t$, $H_t$, $G_t$, $Q_t$, and $R_t$ do not depend on $t$. This is probably the most common application of the Kalman filter. However, when we derive the Kalman filter, we'll see that the extension to time-varying systems does not make things much more difficult. A final point is that we have tacitly assumed that the noise vectors $u_t$ and $v_t$ are independent of each other as well as independent across time.

## 6.2 Kalman Filter: Derivation

The goal of the Kalman filter is to derive the optimum estimate of the state vector $x_t$ given the observations $y_0, y_1, \ldots, y_t$. Since this is a linear, Gaussian environment, we know that the posterior distribution is a Gaussian. So we will derive the conditional expectation $E[x_t|y_0, \ldots, y_t]$, which is the MMSE estimate. We will also derive the covariance of the posterior. In addition to characterizing the variance of the estimation error, the posterior covariance is crucial for computing the conditional mean at each step.

The Kalman filter has two steps. In the **time update**, we estimate the state vector at the *next* time step before seeing the next observation. We define the estimate and error covariance for the time update as follows:

$$\hat{x}_{t+1|t} = E[x_{t+1}|y_0, \ldots, y_t] \tag{6.11}$$

$$\Sigma_{t+1|t} = E[(\hat{x}_{t+1|t} - x_{t+1})(\hat{x}_{t+1|t} - x_{t+1})^T|y_0, \ldots, y_t]. \tag{6.12}$$

The subscript notation $t+1|t$ means that we are making an estimate of the state at $t+1$ given observations all the way up to time $t$. The time update is a *prediction* of the state $x_{t+1}$. This prediction can be useful in its own right; imagine trying to keep a camera pointed at a target between measurements. More important, however, we will use the time update as an intermediate step in computing the full estimate of the state vector from the measurement at time $t+1$. Indeed, the next step is the **measurement update**, which has estimate and error covariance

$$\begin{align}
\hat{x}_{t+1|t+1} &= E[x_{t+1}|y_0,\ldots,y_{t+1}] \tag{6.13}\\
\Sigma_{t+1|t+1} &= E[(\hat{x}_{t+1|t}-x_{t+1})(\hat{x}_{t+1|t}-x_{t+1})^T|y_0,\ldots,y_{t+1}]. \tag{6.14}
\end{align}$$

The beauty of the Kalman filter is that we can update these estimates recursively: rather than computing the conditional expectations from scratch, and storing all of the observations $y_0,\cdots,y_t$ in memory, we need only to store the estimate and estimate covariance from the previous time step. This is effectively an application of Bayes rule: we treat the estimate and error covariance as a Gaussian prior, and the measurement and update steps result from computing the (Gaussian) posterior distribution. Let's look at how this works for each step.

**Time update:** The time update step requires us to solve $\hat{x}_{t+1|t} = E[x_{t+1}|y_0,\ldots,y_t]$. The crucial fact here is that we already know the posterior distribution of $x_t$ given all of the observations up to time $t$; it's a Gaussian with mean $\hat{x}_{t|t}$ and covariance $\Sigma_{t|t}$. Therefore, we can write the estimate as

$$\begin{align}
\hat{x}_{t+1|t} &= E[x_{t+1}|y_0,\ldots,y_t] \tag{6.15}\\
&= E[F_t x_t + G_t u_t|y_0,\ldots,y_t] \tag{6.16}\\
&= F_t E[x_t|y_0,\ldots,y_t] \tag{6.17}\\
&= F_t \hat{x}_{t|t}, \tag{6.18}
\end{align}$$

where the third equality is true because the model noise $u_t$ is zero mean. In other words, to predict $x_{t+1}$ given our previous estimate, we just put it through the dynamic model, and neglect the zero mean noise. Similarly, we can derive the error covariance:

$$\begin{align}
\Sigma_{t+1|t} &= E[(\hat{x}_{t+1|t}-x_{t+1})(\hat{x}_{t+1|t}-x_{t+1})^T|y_0,\ldots,y_t] \tag{6.19}\\
&= E[(F_t(\hat{x}_{t|t}-x_t)+G_{t+1}u_{t+1})(F_t(\hat{x}_{t|t}-x_t)+G_{t+1}u_{t+1})^T|y_0,\ldots,y_t] \tag{6.20}\\
&= E[F_t(\hat{x}_{t|t}-x_t)(\hat{x}_{t|t}-x_t^T F_t^T|y_0,\ldots,y_t] + E[G_{t+1}u_{t+1}u_{t+1}^T G_{t+1}^T|y_0,\ldots,y_t] \tag{6.21}\\
&= F_t\Sigma_{t|t}F_t^T + G_{t+1}Q_{t+1}G_{t+1}^T. \tag{6.22}
\end{align}$$

In other words, the error covariance of the measurement update has two parts: the error covariance of the previous step, put through the dynamics, plus the error covariance of the model noise. The model noise increases the uncertainty of the estimate. Indeed, we could continue to make further predictions using these equations, and the error covariance would continue to grow. The error covariance will decrease only when we take a measurement step.

**Measurement update:** The measurement update requires us to solve $\hat{x}_{t+1|t+1} = E[x_{t+1}|y_0,\ldots,y_{t+1}]$. Here we use the time update as a prior: Given all of the observations up to $y_t$, the state vector $x_{t+1}$ is Gaussian with mean $\hat{x}_{t+1|t}$ and covariance $\Sigma_{t+1|t}$. To incorporate the final measurement $y_{t+1}$, it turns out we only need to make a Bayes rule update of a form we've already seen. The measurement model says that

$$y_{t+1} = H_{t+1}x_{t+1} + v_{t+1}. \tag{6.23}$$

We've already seen this situation before, when we looked at Bayesian estimation of multivariate Gaussian sources. Applying the formula to this measurement model, and using the prior associated with $\hat{x}_{t+1|t}$, we get the MMSE estimate

$$\hat{x}_{t+1|t+1} = \hat{x}_{t+1|t} + \Sigma_{t+1|t} H_{t+1}^T (H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_{t+1})^{-1} (y_{t+1} - H_{t+1} \hat{x}_{t+1|t}), \tag{6.24}$$

and the resulting error covariance

$$\Sigma_{t+1|t+1} = \Sigma_{t+1|t} - \Sigma_{t+1|t} H_{t+1}^T (H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_{t+1})^{-1} H_t \Sigma_{t+1|t}. \tag{6.25}$$

It is common to define $K_{t+1} \triangleq \Sigma_{t+1|t} H_{t+1}^T (H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_t)^{-1}$ as the **Kalman gain matrix**. Essentially the Kalman gain defines a trade-off between the estimate derived from the time update and the estimate that we'd get by looking at the observations alone. One way to think of the Kalman filter is as a linear combination of the estimates given by the dynamic model and the observation model. In true Bayesian fashion, the Kalman gain balances the two models by looking at how confident we are in the time update estimate (as indicated by $\Sigma_{t+1|t}$) and the amount of measurement noise (as indicated by $R_t$). It's easy to convince yourself that if the measurement noise covariance is large, the Kalman filter reduces to repeated time update steps, which simply "propagate" the model dynamics. If the measurement noise is small, the Kalman filter reduces to repeated pseudo-inverses of the observation matrix $H_t$.

Let's put all of this together. A final consideration is how to initialize the Kalman filter: what should the estimate be before we see any measurements at all? A fortunate aspect of sequential measurement is that the effects of initialization usually are "swamped out" by Kalman updates rather quickly. Except in cases where the system is not observable or the measurement noise is obscenely large, the choice of the initialization point is not too critical. We will leave these choices, denoted by $\hat{x}_{0|0}$ and $\Sigma_{0|0}$, fully general, but a common choice is to let the mean be zero and the covariance be the identity matrix.

Regardless of the initialization point, we can boil the Kalman filter down to a few steps:

1. Initialize the filter with estimate $\hat{x}_{0|0}, \Sigma_{0|0}$. Then, for every $t = 1, 2, \cdots$:
2. Compute the time update:

$$\hat{x}_{t|t-1} = F_{t-1} \hat{x}_{t-1|t-1} \tag{6.26}$$

$$\Sigma_{t|t-1} = F_{t-1} \Sigma_{t-1|t-1} F_{t-1}^T + G_{t-1} Q_{t-1} G_{t-1}^T. \tag{6.27}$$

3. Compute the Kalman gain:

$$K_t \triangleq \Sigma_{t|t-1} H_t^T (H_t \Sigma_{t|t-1} H_t^T + R_t)^{-1} \tag{6.28}$$

4. Compute the measurement update:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (y_t - H_t \hat{x}_{t|t-1}) \tag{6.29}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1}. \tag{6.30}$$

The Kalman equations are easy to write down, and not that difficult to derive, but there is much, much more that one can say about this type of estimation problem. First of all, one might wonder if there is any sense in which one can derive a better estimator. In the linear, Gaussian case, there simply isn't! The Kalman filter derives the true posterior, from which the optimum MMSE estimate is found; in the Gaussian setting there is simply no other estimate to choose from. Furthermore, the Kalman filter can be derived in a non-Bayesian setting, as a maximum-likelihood estimator, a minimum-variance estimator, an optimum *linear* estimator, and so on. Gaussianity and linearity greatly simplify estimation, to the point that no matter how you look at it, the Kalman filter is the best estimator.

If one keeps the linear model but assumes that the noise is non-Gaussian, things get more complicated. It turns out that the Kalman filter still gives the correct mean and covariance of the posterior. Thus, the Kalman filter gives the MMSE estimator as well as its error covariance. However, if the noise is multi-modal or heavy-tailed, the MMSE estimator may not be the best choice.

Finally, if one has a non-linear model, either for the model dynamics or the measurement, the Kalman filter is no longer optimal. This is true even if the noise is Gaussian, because Gaussian random variables put through a nonlinear process are no longer Gaussian. In this case, the optimum estimator is harder to obtain. This is a major challenge, because nonlinear process and measurement models come up in many practical applications. In the next section, we'll look at the **extended Kalman filter**, which is the most popular approach to filtering in non-linear systems.

First, let's derive the Kalman filter for the examples we considered earlier.

**Example 6.1: Brownian Motion.** *Brownian motion is described by the following model and measurement equations:*

$$\begin{aligned} x_{t+1} &= x_t + z_t && (6.31)\\ y_t &= x_t + n_t, && (6.32)\end{aligned}$$

*with $z_t \sim \mathcal{N}(0,\sigma^2)$ and $n_t \sim \mathcal{N}(0,\sigma_n^2)$. It's straightforward to see here that $F_t = 1$, $H_t = 1$, and $G_t = 1$, while $Q_t = \sigma^2$ and $R_t = \sigma_n^2$. Putting this together, we arrive at the time and measurement update equations:*

$$\begin{aligned} \hat{x}_{t|t-1} &= \hat{x}_{t-1|t-1} && (6.33)\\ \Sigma_{t|t-1} &= \Sigma_{t-1|t-1} + \sigma^2, && (6.34)\end{aligned}$$

*and*

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + \frac{\Sigma_{t|t-1}}{\Sigma_{t|t-1} + \sigma_n}(y_t - \hat{x}_{t|t-1}) && (6.35)\\ \Sigma_{t|t} &= \Sigma_{t|t-1} - \frac{\Sigma_{t|t-1}^2}{\Sigma_{t|t-1} + \sigma_n} = \frac{\Sigma_{t|t-1}\sigma_n^2}{\Sigma_{t|t-1} + \sigma_n} && (6.36)\end{aligned}$$

*In this simple case, we can easily combine the time and measurement updates into a single update step:*

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t-1|t-1} + \frac{\Sigma_{t-1|t-1} + \sigma^2}{\Sigma_{t-1|t-1} + \sigma^2 + \sigma_n}(y_t - \hat{x}_{t-1|t-1}) && (6.37)\\ \Sigma_{t|t} &= \frac{(\Sigma_{t-1|t-1} + \sigma^2)\sigma_n^2}{\Sigma_{t-1|t-1} + \sigma^2 + \sigma_n^2} && (6.38)\end{aligned}$$

*An interesting exercise is to look at the steady-state of this simple Kalman filter. What is the variance of the error as $t \to \infty$? Let $\Sigma_\infty$ denote the steady-state error variance; we can solve for it* *by plugging it into the recursion for $\Sigma_{t|t}$:*

$$\Sigma_\infty = \frac{(\Sigma_\infty + \sigma^2)\sigma_n^2}{\Sigma_\infty + \sigma^2 + \sigma_n^2}. \tag{6.39}$$

*Rearranging, we end up with the quadratic:*

$$\Sigma_\infty^2 + \Sigma_\infty \sigma^2 - \sigma^2 \sigma^n = 0, \tag{6.40}$$

*which yields the solution*

$$\Sigma_\infty = \frac{\sqrt{\sigma^4 + 4\sigma^2\sigma_n^2} - \sigma^2}{2}. \tag{6.41}$$

*In general, we see that if either the measurement or process noise is small, the overall error variance is small. This is reasonable: if the measurement error is negligible, then we have very accurate state estimates just from the measurement; if the process noise is negligible, the state moves so slowly that we can average out the noise of many measurements to get an accurate estimate. We can go further and find out what the steady-state update equations look like. Combining the time and measurement updates, we get*

$$\hat{x}_{t|t} = \hat{x}_{t-1|t-1} + \frac{\sqrt{\sigma^4 + 4\sigma^2\sigma_n^2}}{\sigma_n^2 + \sqrt{\sigma^4 + 4\sigma^2\sigma_n^2}}(y_t - \hat{x}_{t-1|t-1}). \tag{6.42}$$

*For time-invariant systems, a valid—and common–choice for the initial covariance $\Sigma_0$ is the steady-state covariance. In that case, the Kalman updates are the same every time step, because the error covariance does not change in time. This simplifies the filtering process, as one needs not keep track of the previous error covariance. However, the Kalman filter may not converge on a steady state, as we will see in the following example.*

Let's generalize Brownian motion to a general scalar linear system.

**Example 6.2: Scalar Linear System.** *Consider the following state space model:*

$$x_{t+1} = f x_t + u_t \tag{6.43}$$
$$y_t = h x_t + v_t, \tag{6.44}$$

*where $u_t \sim \mathcal{N}(0, q)$ and $v_t \sim \mathcal{N}(0, r)$ are the process and measurement noises, respectively. Putting this into the update equations, we get*

$$\hat{x}_{t|t-1} = f\hat{x}_{t-1|t-1} \tag{6.45}$$
$$\Sigma_{t|t-1} = f^2 \Sigma_{t-1|t_1} + q, \tag{6.46}$$

*and*

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + \frac{\Sigma_{t|t-1} h}{h^2 \Sigma_{t|t-1} + r}(y_t - h\hat{x}_{t|t-1}) \tag{6.47}$$
$$\Sigma t|t = \Sigma_{t|t-1} - \frac{\Sigma_{t|t-1}^2 h^2}{h^2 \Sigma_{t|t-1} + r} = \frac{r\Sigma_{t|t-1}}{h^2 \Sigma_{t|t-1} + r}. \tag{6.48}$$

*Once again, we can combine the two updates into a single recursion:*

$$\hat{x}_{t|t} = f\hat{x}_{t-1|t-1} + \frac{(f^2\Sigma_{t-1|t-1} + q)h}{h^2(f^2\Sigma_{t-1|t-1} + q) + r}(y_t - hf\hat{x}_{t|t-1}) \tag{6.49}$$

$$\Sigma_{t|t} = \frac{rf^2\Sigma_{t-1|t-1} + rq}{h^2f^2\Sigma_{t-1|t-1} + h^2q + r}. \tag{6.50}$$

*Here again, we can solve for the steady-state covariance. Solving the associated quadratic gives us*

$$\Sigma_\infty = \frac{1}{2}\left((\frac{r}{h^2}(1-f^2) - q)^2 + \frac{4rq}{h^2}\right) - \frac{r}{2h^2}. \tag{6.51}$$

*Again, this expression makes sense. The larger the noises or the gain $f$, the harder it is to keep track of the system, and the larger the steady-state variance. Here, however, we need to be careful with the steady-state variance. It turns out that we are only guaranteed that the variance converges at all if $|f| < 1$. Otherwise, the system is unstable, and the Kalman filter may not converge.*

Let's return to the simple Kinematic example. However, in order to simplify the analysis, let's consider a one-dimensional reduction of it.

**Example 6.3: One-dimensional Kinematics.** *The state space model is*

$$x_{t+1} = \begin{bmatrix} p_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta \end{bmatrix} u_t \tag{6.52}$$

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + v_t, \tag{6.53}$$

*where $u_t \sim \mathcal{N}(0, q)$ and $v_t \sim \mathcal{N}(0, r)$. Considering the state vector only, the time update equation is simply*

$$\hat{x}_{t|t-1} = \begin{bmatrix} \hat{p}_{t-1|t-1} + \Delta\hat{v}_{t-1|t-1} \\ \hat{v}_{t-1|t-1} \end{bmatrix}, \tag{6.54}$$

*and the measurement update is*

$$\hat{x}_{t|t} = \begin{bmatrix} \hat{p}_{t|t-1} \\ \hat{v}_{t|t-1} \end{bmatrix} + K_t(y_t - \hat{p}_{t|t-1}), \tag{6.55}$$

*where in this case the Kalman gain is a $2 \times 1$ matrix that we won't evaluate explicitly owing to the need to perform a matrix inverse. It is more challenging to compute the steady-state of this system. In practice, the Kalman gain matrix—which varies in time—is often replaced with a time-invariant "filter", resulting in the measurement update equation*

$$\hat{x}_{t|t} = \begin{bmatrix} \hat{p}_{t|t-1} \\ \hat{v}_{t|t-1} \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta/\Delta \end{bmatrix} (y_t - \hat{p}_{t|t-1}), \tag{6.56}$$

*where $\alpha, \beta$ are constants that trade-off between response speed (higher values give more weight to the measurements) and estimation accuracy (smaller values filter out the measurement noise more aggressively). A good exercise is to show that the values that minimize the mean squared error in the asymptote correspond to the steady-state Kalman gain.*

A final note on convergence. In general, we want the estimation error and the resulting Kalman gain to converge to a steady state. Otherwise the estimate may diverge far from the true value, and the estimate may oscillate wildly with new measurements. One sufficient—but not necessary—condition is that the eigenvalues of $F$ are all smaller than one in magnitude. This is equivalent to saying that the linear system is stable. Indeed, any stable system can be observed in the limit. A more precise criterion is that of *observability*, a topic familiar in control theory. The upshot is that an observable system is one where, if we eliminate the noise on the system and measurements, we can uniquely determine the $n$-dimensional state vector from $n$ successive measurements $y_t$. Specifically, if $F$ and $H$ are constant in time, then we can build the **observability matrix**:

$$O = \begin{bmatrix} H \\ HF \\ HF^2 \\ \cdots \\ HF^{n-1} \end{bmatrix}. \tag{6.57}$$

This matrix represents the first $n$ observations of the linear filter. If $O$ is invertible, then it is possible to recover the states $x_1, \ldots, x_n$ from the outputs $y_1, \ldots, y_n$, up to the noise on the system and the observations. When this condition holds, we say that the system is **observable**, and the Kalman filter converges to a steady-state.

## 6.3 The Extended Kalman Filter

What if the equations describing the state update or the measurements are nonlinear? For example, the measurements of Doppler radar and sonar systems are nonlinear in the position and velocity of the object being tracked. Furthermore, non-linear system dynamics are common in engineering problems, from aircraft dynamics to chemical processes. This poses a challenge for tracking. Nonlinear dynamics destroy Gaussianity: even if the model and measurement noise are Gaussian, inputting them to a nonlinear system results in a non-Gaussian output. The mean and covariance of the state are no longer enough to characterize the entire distribution; even worse, we cannot easily compute the mean and covariance resulting from non-linear dynamics. Optimum tracking in a nonlinear environment it seems, is a hopeless task.

Of course, engineers have been tackling hopeless tasks since time immemorial, and there are multiple successful approaches to non-linear filtering. We will look at the simplest and (probably) most popular, the **extended Kalman filter** (EKF). The punchline of the extended Kalman filter to take a Taylor series expansion of the nonlinear system at each step. From the Taylor series, we can build a linear approximation of the system, and we can use the Kalman filter equations with only a few modifications. As long as the system is not "too" non-linear, the Taylor approximation models the evolution of the state fairly well, and the estimation quality is high. However, the extended Kalman filter is not guaranteed to converge precisely due to the nonlinearities. We cannot guarantee that the linear approximation is sufficiently close to the true system, so we cannot guarantee that the EKF provides an estimator that is sufficiently close to the optimum.

Let's make all of this concrete. Again we will suppose we have a sequence of state vectors $x_t \in \mathbb{R}^d$ and a sequence of corresponding observations $y_t$. However, we have a nonlinear set of equations governing their relationship:

$$x_{t+1} = f_t(x_t) + G_t u_t \tag{6.58}$$

$$y_t = h_t(x_t) + v_t, \tag{6.59}$$

where $f_t : \mathbb{R}^d \to \mathbb{R}^d$ is a nonlinear function mapping states at time $t$ to time $t+1$ and $h_t : \mathbb{R}^d \to \mathbb{R}^n$ is a function mapping states to observations. The terms $u_t$, and $v_t$, and $G_t$ are defined as before: Gaussian noises with known covariances $Q_t$ and $R_t$, and a known matrix $G_t$ that maps the model noise to changes in the state vector. The standard linear framework is a special case, which we can see by letting $f_t = F_t x_t$ and $h_t = H_t x_t$.

The extended Kalman filter is based on the notion of linearizing a function about an operating point. For a scalar function $f(x)$, we know from Taylor's theorem that $f(x) \approx f(x_0) + d/dx f(x_0)(x - x_0)$, for $x$ sufficiently close to $x_0$. We can generalize this to vector functions by recalling the **Jacobian** matrix from multi-variable calculus. Let $f : \mathbb{R}^d \to \mathbb{R}^m$. Then, the Jacobian is

$$J_f(x_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_d}. \end{bmatrix} \tag{6.60}$$

Then, the approximation $f(x) \approx f(x_0) + [J_f(x_0)](x - x_0)$ holds, where $[J_f(x_0)](x - x_0)$ is the matrix-vector multiplication of $x - x_0$ and the Jacobian matrix. The Jacobian gives us a linear transformation that approximates the nonlinear dynamics given above. In fact, given a reference point $x_0$, we can write the approximate dynamics as follows. Define

$$F_t(x_0) \triangleq J_{f_t}(x_0) \tag{6.61}$$

as the Jacobian of the nonlinear function $f_t$ around the point $x_0$. Then, the dynamics equation is approximated by

$$x_{t+1} \approx f_t(x_0) + F_t(x_0)(x_t - x_0) + G_t u_t. \tag{6.62}$$

We can also approximate the measurement model. Define

$$H_t(x_0) \triangleq J_{h_t}(x_0) \tag{6.63}$$

as the Jacobian of the nonlinear function $f_t$ around the point $x_0$. Then, the measurement equation is approximated by

$$y_t \approx h_t(x_0) + H_t(x_0)(x_t - x_0) + v_t. \tag{6.64}$$

Using this approximate dynamics, we can derive a set of time and measurement updates that mimic the Kalman filter. As before we define $\hat{x}_{t|t-1}$ and $\Sigma_{t|t-1}$, which describe the estimate of $x_t$ given measurements up to time $t-1$, and $\hat{x}_{t|t}$ and $\Sigma_{t|t}$, which describe the estimate of $x_t$ given measurements up to time $t$. We form these estimates recursively, using the time and measurement updates, respectively. Let's look at each step in turn.

**Time update:** The first step is to construct an estimate of $x_t$ given all of the previous measurements, which amounts to predicting the state one step in the future. Here, we will use the nonlinear dynamics directly. Because the model noise is zero mean, the best estimate of the state is just the previous state estimate put through the nonlinear dynamics:

$$\hat{x}_{t|t-1} = f_{t-1}(\hat{x}_{t-1|t-1}). \tag{6.65}$$

However, it's a challenge to compute the covariance of the new estimate, because the process does not preserve the Gaussianity of the state vector. Instead, we use the linear approximation to derive an approximate covariance. To find the best approximation, we linearize about the best estimate so far, which is $\hat{x}_{t-1|t-1}$. Indeed, define the matrix $F_{t-1} \triangleq F_{t-1}(\hat{x}_{t-1|t-1})$ to shorthand the linearization around the previous estimate. Then, we update the covariance estimate using the standard Kalman update:

$$\Sigma_{t|t-1} = F_{t-1}\Sigma_{t-1|t-1}F_{t-1}^T + G_{t-1}Q_{t-1}G_{t-1}^T. \tag{6.66}$$

It should be stressed that this update does not produce the "true" error covariance. The state is no longer Gaussian, and this is no longer its true covariance. This is merely an approximate expression.

**Measurement update:** Next, we derive the measurement update. The first step is to compute the Kalman gain, which requires us to linearize the measurement dynamics. Here, too, we want to use the best measurement possible, which now is the result of the time update. Thus, define $H_t = H_t(\hat{x}_{t|t-1})$. We can define the Kalman gain almost exactly as before

$$K_t = \Sigma_{t|t-1}H_t^T(H_t\Sigma_{t|t-1}H_t^T + R_t)^{-1}. \tag{6.67}$$

Then, we get the following measurement update equations:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - h_t(\hat{x}_{t|t-1})) \tag{6.68}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1}. \tag{6.69}$$

Notice that we use the true nonlinear dynamics in the first equation. The basic idea of the extended Kalman filter is to use the nonlinear dynamics directly whenever you can, and when you can't, use the best possible linearization. When making the time update and when correcting for the measurement, we can evaluate $f_{t-1}(\hat{x}_{t-1|t-1})$ and $h_t(\hat{x}_{t|t-1})$ directly; there is no reason to approximate them, so why do it? However, in order to compute a covariance, we need a linear approximation, so we use the Jacobian matrices $F_t$ and $H_t$, evaluated at the latest possible measurement.

Putting it all together, the extended Kalman filter reduces to the following steps:

1. Initialize the filter with estimate $\hat{x}_{0|0}, \Sigma_{0|0}$. Then, for every $t = 1, 2, \cdots$:

2. Compute the time update:

$$\hat{x}_{t|t-1} = f_{t-1}(\hat{x}_{t-1|t-1}) \tag{6.70}$$

$$F_{t-1} \triangleq J_{f_{t-1}}(\hat{x}_{t-1|t-1}) \tag{6.71}$$

$$\Sigma_{t|t-1} = F_{t-1}\Sigma_{t-1|t-1}F_{t-1}^T + G_{t-1}Q_{t-1}G_{t-1}^T. \tag{6.72}$$

3. Compute the Kalman gain:

$$H_t \triangleq J_{h_t}(\hat{x}_{t|t-1}) \tag{6.73}$$

$$K_t \triangleq \Sigma_{t|t-1}H_t^T(H_t\Sigma_{t|t-1}H_t^T + R_t)^{-1} \tag{6.74}$$

4. Compute the measurement update:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - h_t(\hat{x}_{t|t-1})) \tag{6.75}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1}. \tag{6.76}$$

It is usually wise to choose the prior $\Sigma_{0|0}$ to encode large variances. Because we can incorporate the dynamics only approximately, it makes sense to err on the side of trusting the measurements. Indeed, the trade-off between trusting the model dynamics and trusting the measurements is sub-optimum for the EKF. The resulting estimator works well in practice, but it has no guarantees to being MMSE or even of converging. Indeed, because we linearize the system about the dynamic trajectory of estimates $\hat{x}_{t|t}$, we cannot say anything about the quality of the approximation in advance.

Let's look at a reformulated version of the simple kinematics example we've been looking at.

**Example 6.4: Non-linear Kinematics.** *We consider a two-dimensional version of the constant-acceleration kinematics we've looked at earlier. We suppose that the discrete-time dynamics is the same linear model that we've been looking at all chapter, but in two dimensions. In particular, the state vector is* $x_t = (p_t^x, p_t^y, v_t^x v_t^y)$*, and the dynamics follows*

$$\begin{bmatrix} p_t^x \\ p_t^y \\ v_t^x \\ v_t^y \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t+1}^x \\ p_{t+1}^y \\ v_{t+1}^x \\ v_{t+1}^y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_t^x \\ u_t^y \end{bmatrix}. \tag{6.77}$$

*Equivalently, this follows the standard linear dynamics, with*

$$F = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}. \tag{6.78}$$

*On the other hand, the measurement dynamics is non-linear; specifically,*

$$h_t(x_t) = \begin{bmatrix} \sqrt{(p_t^x)^2 + (p_t^y)^2} \\ \arctan \frac{p_t^y}{p_t^x} \end{bmatrix} \quad R = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}. \tag{6.79}$$

*In other words, we get a noisy measurement of the range and the bearing to the object. This is an approximation of, for example, radar systems. We don't get a position reading directly from radar measurements, but instead learn in what direction and how far away the object is.*[1] *To derive the update equations for this model, we need to compute the Jacobian. Recalling that* $d/dx \arctan x = \frac{1}{1+x^2}$*, we get that the Jacobian is*

$$J_{h_t}(x_t) = \begin{bmatrix} \frac{p_t^x}{\sqrt{(p_t^x)^2+(p_t^y)^2}} & \frac{p_t^y}{\sqrt{(p_t^x)^2+(p_t^y)^2}} & 0 & 0 \\ -\frac{p_t^y}{(p_t^x)^2+(p_t^y)^2} & \frac{p_t^x}{(p_t^x)^2+(p_t^y)^2} & 0 & 0 \end{bmatrix}. \tag{6.80}$$

*The measurement does not depend on the velocity of the object, so the last two columns are zero. The first two columns are just the derivatives of the measurements with respect to the* $x$ *and* $y$ *coordinates, respectively.*

*The model dynamics are linear for this case, so the time update is exactly as in the standard Kalman filter:*

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1} \tag{6.81}$$

$$\Sigma_{t|t-1} = F\Sigma_{t-1|t-1}F^T + Q, \tag{6.82}$$

[1] We could look even more directly at the radar measurement. Doppler radar, for example, provides delay and

*with the matrices $F$ and $Q$ as defined above. On the other hand, we use the linearization of the measurement model about $\hat{x}_{t|t-1}$ in order to compute the Kalman gain and the measurement update:*

$$\begin{aligned} H_t &\triangleq J_{h_t}(\hat{x}_{t|t-1}) \\ K_t &\triangleq \Sigma_{t|t-1} H_t^T (H_t \Sigma_{t|t-1} H_t^T + R_t)^{-1} \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t(y_t - h_t(\hat{x}_{t|t-1})) \\ \Sigma_{t|t} &= \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1}. \end{aligned}$$

We have glossed over a few important issues with tracking and filtering. Each of these warrants examination by anyone studying tracking and filtering:

- More sophisticated approaches to nonlinear filtering. Perhaps the most popular approach in recent years is the **unscented Kalman filter**. Rather than computing a linearization of the system, the UKF takes a few points from the Gaussian distribution of the state at time $t$. These points are put into the nonlinear dynamics directly; from the output, one computes a sample covariance, which is used to update the estimate quality and compute the Kalman gain. This has the benefit of tracking more severe nonlinearities in the state or measurement model. On the other hand, it still relies on a Gaussian approximation of the state, which does not hold for nonlinear systems.

- **Particle filtering.** This is a modern, sophisticated approach to non-linear filtering. Rather than take a few samples from a Gaussian approximation, particle filters keep a large collection of samples from an arbitrary distribution. These samples—called "particles"; hence the name!—are put into the nonlinear dynamics; the output is used to sample from the posterior distribution induced by the nonlinear dynamics. As the number of particles goes to infinity, the particle filter converges on the "true" posterior at every time step, from which one can extract an MAP or MMSE estimate. On the other hand, the particle filter is computationally expensive due to the need to maintain many particles. As a result, it is less common in practical applications.

- Smoothing. We have supposed that we want to make an updated measurement at each time $t$, using all of the measurement information available to us. However, what if we are willing to wait a little bit longer? After all, a measurement at time $t+1$ tells us something about the state at time $t$. The **Rauch-Tung-Striebel** (RTS) **smoother** is an approach to using an entire batch of data to estimate jointly a trajectory through state space. The RTS smoother is a two-pass algorithm: the first pass is essentially a(n extended) Kalman filter all the way to the final measurement. Then, the second pass works backward, using non-causal information to refine the state estimate. In the linear/Gaussian case, the RTS smoother gives the MMSE estimate of any state vector $x_t$ given *all* of the data available.

Doppler frequency readings, from which one can infer the distance and velocity of the object. These relationships are non-linear. We won't go into it in these notes, but it is relatively straightforward to derive the EKF for this system.