Basics of Machine Learning

Martha White

January 1, 2023

Table of Contents

Notation Reference

Set notation

$\mathcal{X}$ A generic set of values. For example, $\mathcal{X} = \{0, 1\}$ is the set containing only 0 and 1, $\mathcal{X} = [0, 1]$ is the interval from 0 to 1 and $\mathcal{X} = \mathbb{R}$ is the set of real numbers. Depending on occasion, symbols such as A, B, Ω, and others will also be used as sets.

$\mathcal{P}(\mathcal{X})$ The power set of $\mathcal{X}$, a set containing all possible subsets of $\mathcal{X}$.

$[a, b]$ Closed interval with $a < b$, including both a and b.

(a, b) Open interval with $a < b$, with neither a nor b in the set.

$(a, b]$ Open-closed interval with $a < b$, including b but not a.

$[a, b)$ Closed-open interval with $a < b$, including a but not b.

Tuples, Vectors and Matrices

x Unbold lowercase variables are generally scalars.

$a_1, a_2, \ldots, a_m$ A sequence of m items.

$(x_1, x_2, \ldots, x_d)$ A tuple; i.e., an ordered list of d elements. When $(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, the tuple will be treated as a column vector $\mathbf{x} = [x_1\ x_2\ \ldots\ x_d]^\top$.

$\mathbf{x}$ Bold lowercase variables are vectors. By default, vectors are column vectors.

$\mathbf{X}$ Bold uppercase variables are matrices. A matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a two-dimensional array with n rows and d columns. This bold variable looks like a multivariate random variable, $\boldsymbol{X}$, but the random variable is italicized. It will often be clear from context when this is a multivariate random variable and when it is a matrix.

$\mathbf{X}^\top$ The transpose of the matrix, where we swap the elements around the diagonal of the matrix. An $n \times d$ matrix consisting of n vectors each of dimension d can be expressed as

$$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n]^\top .$$

A vector $\mathbf{x}$ is a matrix (a $d \times 1$ matrix), and the transpose similarly flips the orientation: a row vector becomes a column vector, and a column vector becomes a row vector.

$\langle \mathbf{a}, \mathbf{b} \rangle$ We primarily use the transpose to obtain the **dot product** between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^\top \mathbf{b} = \sum_{j=1}^{d} a_j b_j.$$

Function notation

$f : \mathcal{X} \to \mathcal{Y}$ The function is defined on domain $\mathcal{X}$ to co-domain $\mathcal{Y}$, taking values $x \in \mathcal{X}$ and sending them to $f(x) \in \mathcal{Y}$.

$\frac{df}{dx}(x)$ The derivative of a function at $x \in \mathcal{X}$, where $f : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X} \subset \mathbb{R}$.

$\nabla f(\mathbf{x})$ The gradient of a function at $\mathbf{x} \in \mathcal{X}$, where $f : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X} \subset \mathbb{R}^d$. It holds that

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_d} \right).$$

$\min_{a \in \mathcal{B}} c(a)$ The minimum value of a function c across values a in a set $\mathcal{B}$. Note that this is equivalent to $\max_{a \in \mathcal{B}} -c(a)$.

$\operatorname{argmin}_{a \in \mathcal{B}} c(a)$ The item a in set $\mathcal{B}$ that produces the minimum value $c(a)$. Note that this is equivalent to $\operatorname{argmax}_{a \in \mathcal{B}} -c(a)$.

$c : \mathbb{R}^d \to \mathbb{R}$ A generic objective function, that we want to minimize, for the learned variable $\mathbf{w}$. This could be, for example, a loss plus a regularizer.

Random variables and probabilities

X A univariate random variable is written in uppercase.

$\mathcal{X}$ The space of values for the random variable.

x Lowercase variable is an instance or outcome, $x \in \mathcal{X}$.

$\boldsymbol{X}$ A multivariate random variable is written bold uppercase.

$\mathbf{x}$ Lower case bold variable is a multivariate instance, $\mathbf{x} \in \mathcal{X}$. Note that we use the set $\mathcal{X}$, for both univariate and multivariate random variables. We explicitly distinguish between univariate and multivariate outcomes because it changes whether we use a simple product or dot products.

$\mathcal{N}(\mu, \sigma^2)$ A univariate Gaussian distribution, with parameters μ, σ^2.

$\sim$ indicates that a variable is distributed as e.g., $X \sim \mathcal{N}(\mu, \sigma^2)$.

Parameters and estimation

$\mathcal{D}$ A data set, typically composed of n elements of multivariate inputs $\mathbf{X} \in \mathbb{R}^{n \times d}$ and univariate outputs $\mathbf{y} \in \mathbb{R}^n$ or multivariate outputs $\mathbf{Y} \in \mathbb{R}^{n \times m}$. The data set will also be referred to as a set of indexed tuples; i.e., $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\}$.

$\mathcal{F}$ The *function class* or *hypothesis space*. Our learning algorithms will be restricted implicitly to selecting a function from this set. For example, in linear regression, our function class is $\mathcal{F} = \{f : \mathbb{R}^d \to \mathbb{R} \mid f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} \text{ for some } \mathbf{w} \in \mathbb{R}^d\}$. We may overload this and write $\mathcal{F} = \{\mathbf{w} \in \mathbb{R}^d\}$, since the set of weights defines this function class.

$\boldsymbol{\omega}$ The true parameters for the (generalized) linear regression and classification models, typically with $\boldsymbol{\omega} \in \mathbb{R}^d$.

$\mathbf{w}$ The approximated parameters for the (generalized) linear regression and classification models, typically with $\mathbf{w} \in \mathbb{R}^d$.

$\mathbf{w}_{\text{MLE}}(\mathcal{D})$ When discussing $\mathbf{w}$ as the maximum likelihood solution on some data, we write $\mathbf{w}_{\text{MLE}}(\mathcal{D})$, to indicate that the variability arises from $\mathcal{D}$.

Norms

$\|\mathbf{x}\|$ A norm on $\mathbf{x}$.

$\|\mathbf{x}\|_2$ The ℓ_2 norm on a vector, $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$. This norm gives the Euclidean distance from the origin of the coordinate system to $\mathbf{x}$; that is, it is the length of vector $\mathbf{x}$.

$\|\mathbf{x}\|_2^2$ The squared ℓ_2 norm on a vector, $\|\mathbf{x}\|_2^2 = \sum_{i=1}^d x_i^2$.

$\|\mathbf{x}\|_p$ The general ℓ_p norm on a vector, $\|\mathbf{x}\|_p = (\sum_{i=1}^d |x|_i^p)^{1/p}$.

Useful formulas and rules

$$\log\left(\frac{x}{y}\right) = \log(x) - \log(y)$$

$$\log\left(x^y\right) = y \log(x)$$

$$\sum_{i=1}^m a_i \int_{\mathcal{X}} f_i(x) p(x) dx = \int_{\mathcal{X}} \sum_{i=1}^m a_i f_i(x) p(x) dx \quad \triangleright \text{ Can bring the sum into the integral}$$

$$\frac{d}{dx} \int_{\mathcal{X}} f(x) p(x) dx = \int_{\mathcal{X}} \frac{d}{dx} f(x) p(x) dx \quad \triangleright \text{ Can (almost always) bring a derivative into an integral}$$

Chapter 1

Introduction

This book focuses on the fundamentals underlying machine learning. In this chapter, you will see the basic problem formulation, and why it is that we formalize the problem using probability. The remainder of the chapter provides a brief refresher of useful background, and concludes with a brief description of the structure of this book.

1.1 A First Step in Machine Learning: Motivating a Probabilistic Formulation

Machine learning involves a broad range of techniques for learning from data. A central goal — and the one we largely discuss in this handbook — is prediction. Many techniques learn a function $f : \mathbb{R}^d \to \mathbb{R}$ that inputs attributes or features about an item, and produces an output prediction about that item. For example, consider a setting where you would like to guess or predict the price of a house based on information about that house. You might have features such as its age, the size of the house and, of course, distance to the nearest bakery. Without any previous examples of house costs, i.e., without any data, it might be hard to guess this price. However, imagine you are given a set of house features and the corresponding selling costs, for houses that sold this year. Let $\mathbf{x} \in \mathbb{R}^d$ be a vector of the features for a house, in this case $\mathbf{x} = [x_1\ x_2\ x_3]$ =[`age, size, distance to bakery`] and the target y = price. If we have 10 examples or instances of previous house prices, we have a dataset: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{10}, y_{10})$, where $(\mathbf{x}_i, y_i)$ is the feature-price pair for the ith house in your set of instances. A natural goal is to find a function f that accurately recreates the data, for example by trying to find a function f that results in a small difference between the prediction, $f(\mathbf{x}_i)$, and the actual price, y_i, for each house.

We can formalize this as an optimization problem. Imagine we have some space of possible functions, $\mathcal{F}$, from which we can select our function f. For a simple case, let us imagine that the function is linear: $f(\mathbf{x}) = w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3$ for any $\mathbf{w} = [w_0\ w_1\ w_2\ w_3] \in \mathbb{R}^d$ where w_0 is the intercept of the linear function. We can try to find a function from the class of linear functions that minimizes these squared differences

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{10} (f(\mathbf{x}_i) - y_i)^2$$

As we will see later, this optimization problem is simple to solve for linear functions. The solution is a straight line that tries to best fit the observed targets y. A simple illustration of such a function, for only one attribute, is depicted in Figure 1.1.

Once we have this function, when we see a new house, we hope that it is similar enough to the previous houses so that this function adequately predicts its house price. The learned

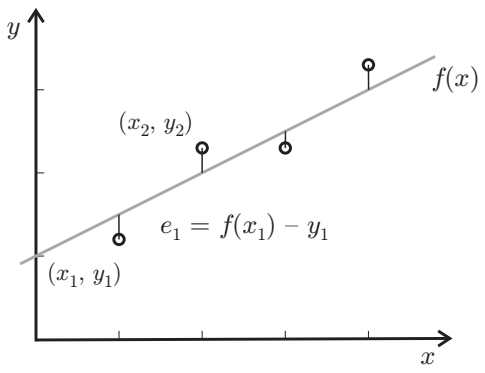


Figure 1.1: An example of a linear function fit to the data set $\mathcal{D} = \{(1, 1.2), (2, 2.3), (3, 2.3), (4, 3.3)\}$*, with errors sum of squared errors* $e_1^2 + e_2^2 + e_3^2 + e_4^2$*.*

function f interpolates between these 10 points to predict on unseen points. But, a natural question is, did we interpolate well and is the learned f going to produce an accurate prediction on new houses? If you want to use this learned function f in practice, you want to have such a characterization.

In the agnostic development above, it is difficult to answer such questions. We can make intuitive modifications that we hope will provide more accurate predictions, like extending the class of functions to complex non-linear functions. But, these functional modifications still do not help characterize accuracy of the prediction on new houses. Rather, what we are missing is a notion of confidence. How confident are we in the predictions? Did we see enough previous houses to be confident about this prediction? What is the source of variability? How do we deal with variability? All these types of questions require a probabilistic treatment.

In this book, we start by providing an introduction to probability, to provide a base for dealing with uncertainty in machine learning. We then return to learning these functions, once we have the probabilistic tools to better understand how to approach the answers to these questions. Much of the required mathematical background will involve basic understanding of probability and optimization; this book will attempt to provide most of that required background throughout.

1.2 A Brief Mathematics Refresher

Machine learning can be well-characterized as a field of applied mathematics. To understand machine learning concepts, it is important to be comfortable with mathematical terminology and concepts. This requirement can seem like a barrier, since you have to learn the language of mathematics simultaneously to the machine learning concepts. This added difficulty is not so uncommon in learning: to learn about Spanish poetry, for example, you may first have to learn the underlying language (Spanish). If you understand that mathematics is just a language in which you are not yet proficient—rather than that you have some inherent inability—then you can embrace this part of machine learning and buckle down and learn the needed language. And what better way to learn this language than immersion in a useful and fun topic: machine learning.

The key concepts to recall for these notes involve basic set notation, function notation and calculus basics. Background on probability and statistics is also critical, but will not be covered in this section, since it has its own dedicated chapter (Chapter 2). After reading this section, look over the notation section at the beginning of this book, which summarizes the formulas here, as well as the notation for the book. Some of the definitions in the notation section will not be addressed in this section, but rather just-in-time when we need to start using it later in the notes.

We will need to reason about sets of items. For example, a set $\mathcal{X} = \{0, 1\}$ is the set containing only 0 and 1, $\mathcal{X} = [0, 1]$ is the interval from 0 to 1 and $\mathcal{X} = \mathbb{R}$ is the set of real numbers. Other symbols, such as Ω and $\mathcal{Y}$, will also be used to denote sets. To write a discrete (or countable) set of items, we use curly brackets { and }, whereas for a continuous (or uncountable) set we use the interval notation with square brackets [and].

For sets with multiple dimensions, we need to specify the set for each dimension. For example, to specify a two-dimensional vector (x, y) with each element in $[0, 1]$, we write

$$[0, 1] \times [0, 1] = [0, 1]^2$$

where $[0, 1]^2$ is a more compact way to write this set. If instead x is from $[-10, 10]$ and y is from $[4, 6.5]$, then we write $[-10, 10] \times [4, 6.5]$. This all extends to more than two variables, where if we have (x, y, z) all from $[-1, 1]$, we write $[-1, 1] \times [-1, 1] \times [-1, 1] = [-1, 1]^3$, or if they are all simply from the reals, we write that the triplet (vector) is from $\mathbb{R}^3$. The two sets also do not have to be of the same type. For example, y could instead be from $\{1, 2, 3\}$ and we would write $[-10, 10] \times \{1, 2, 3\}$. More generically, we may simply consider some items from a set $\mathcal{X}$ and others from a set $\mathcal{Y}$, where their joint space is $\mathcal{X} \times \mathcal{Y}$. For example, we might have vectors $\mathbf{x} \in \mathcal{X}$ where $\mathcal{X} = \mathbb{R}^4$ and scalars $y \in \mathcal{Y}$ where $\mathcal{Y} = [-1, 1]$.

We will often use logarithms and exponentials, because they arise due to the probabilistic formulation we use. We always use base e for the logarithm—distinguished by calling it ln—since we use it as the inverse for the exponential: $f(x) = e^x$ has $f^{-1}(y) = \ln y$ where $f^{-1}(e^x) = \ln(e^x) = x$. To make it easier to read, we will usually write $\exp(x)$ instead of e^x. When we write a function $f : \mathcal{X} \to \mathcal{Y}$, we say that $\mathcal{X}$ is the domain and $\mathcal{Y}$ is the co-domain, where f can only input values x from the domain, and only outputs value $f(x)$ in the codomain. For these functions, $f(x) = e^x$ has $f : \mathbb{R} \to (0, \infty)$ where e^x approaches but does not reach zero, even for very high magnitude negative numbers x. Therefore we write the open set $(0, \infty)$. This set is also written $\mathbb{R}^+$, meaning the set of positive numbers.

Exercise 1: What is the domain and co-domain for the function $f(y) = \ln y$? □

There are a few rules to recall for logarithms and exponentials. These include:

$$\begin{aligned}
&\ln(ab) = \ln(a) + \ln(b) \\
&\ln\left(\frac{a}{b}\right) = \ln(a) - \ln(b) \\
&\ln\left(a^b\right) = b \ln(a) \\
&\exp(a + b) = \exp(a) \exp(b) \\
&\exp(a)^b = \exp(ab)
\end{aligned}$$

Exercise 2: Show the second logarithm rule above, using the first rule. □

Exercise 3: For[1] $b \in \mathbb{N}_1 = \{1, 2, 3, \ldots\}$, prove that $\exp(a)^b = \exp(ab)$ using induction. Start with the base case $b = 1$. □

Exercise 4: For $b \in \mathbb{R}$, prove that $\exp(a)^b = \exp(ab)$. Hint: use the fact that $\exp(a)^b = \exp(\ln(\exp(a)^b))$ because applying the logarithm and then the exponential function does not change the input. □

We will talk about *continuous functions*. Recall that a function is called continuous if it does not have abrupt changes in value, or discontinuities. The functions w^2 and $w + 5$ are both continuous: you can trace your finger continuously along the lines produced by these functions. The function

$$f(w) = \begin{cases} w^2 & w \in [-1, 1] \\ 5 + w & w < -1 \\ 5 - w & w > 1 \end{cases} \tag{1.1}$$

has two discontinuities, one at $w = -1$ and at $w = 1$. The function changes from $f(1) = 1$ and suddenly jumps up to > 5 right after 1. This discontinuous function is depicted in Figure 1.2. Note that the function restricted to particular subintervals is continuous: it is continuous on the interval $w \in [-1, 1]$, on the interval $(-\infty, -1)$ and on the interval $(1, \infty)$.[2]

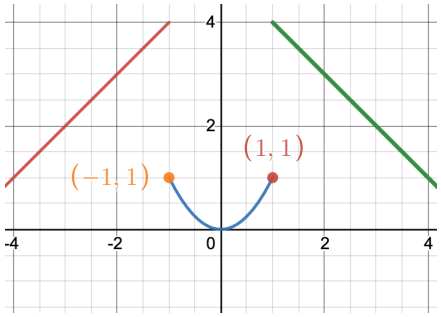


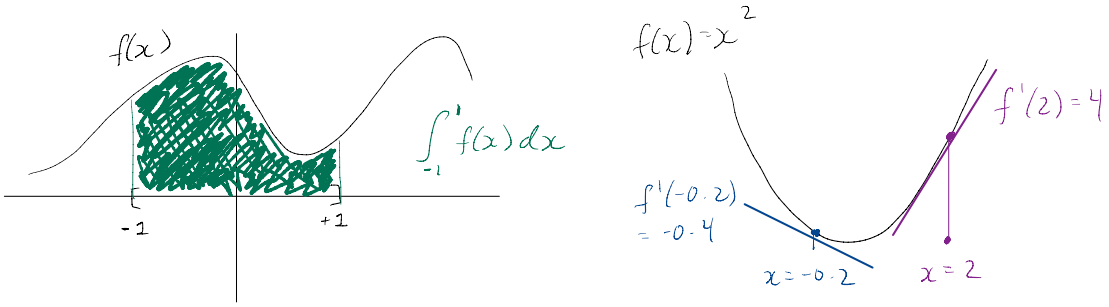
Figure 1.2: An example of a function that is not continuous, with formula in Eq. (1.1).

We will use summation and integration notation frequently. Both summation and integration accumulate the values across sets. When summing a function value $f(x)$ over the elements of a discrete set, such as $\mathcal{X} = \{a, b, c\}$, we write $\sum_{x \in \mathcal{X}} f(x)$. If we are summing from 1 to n, we write $\sum_{i=1}^{n}$. If we accumulate over a continuous set, like the interval $\mathcal{X} = [a, b]$, then we need to use integration, which can be written as

$$\int_{\mathcal{X}} f(x)dx \qquad \text{or} \qquad \int_a^b f(x)dx.$$

[1] For these notes, the natural numbers include zero $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. We explicitly use a subscript $\mathbb{N}_1$ to indicate the natural numbers without zero, starting at one.

[2] We do not need the formal definition of a continuous function, since the intuitive definition is sufficient. For you interest, the formal definition is that a function $f : \mathcal{X} \to \mathbb{R}$ is continuous at $x_0 \in \mathcal{X}$ if for any $\epsilon > 0$ there exists a $\delta > 0$ such that if $\|x - x_0\| < \delta$ for $x \in \mathcal{X}$ then $|f(x) - f(x_0)| < \epsilon$. In other words, for arbitrarily nearby points x and x_0 (δ can be made very small) the difference in function values is also arbitrarily small. The discontinuous function example does not satisfy this property at $x_0 = 1$, because in the neighborhood around 1, if we pick an $\epsilon < 4$, we can pick an arbitrarily small neighborhood (δ approaching zero), but the distance in the function values remains higher than ϵ.



(a) Integral of a function over $[-1, 1]$.

(b) Derivative of $f(x) = x^2$

Figure 1.3: In (a), we see that the integral gives the area under the function in the specified interval, between -1 and 1. In (b), we see that the derivative gives the slope of the line tangent to the function, at the given point. This slope—its sign and magnitude—reflects how quickly the function is increasing or decreasing at that point.

Integration corresponds to the area under a function, for a given input range, as shown in Figure 1.3a. For us, it will be useful for computing probabilities of ranges of outcomes (e.g., what is the probability that the car will stop within 3 to 5 seconds?). We will not need to compute complicated integrals, but it will be useful to remember that constants c come out of integrals and that the definite integral of a constant has a simple form.

$$\int_{\mathcal{X}} cf(x)dx = c\int_{\mathcal{X}} f(x)dx$$
$$\int_a^b cdx = c\left(x|_a^b\right) = c(b-a)$$

For optimization, we will use derivatives. The derivative of a function reflects how that function changes locally. For a function f, the derivative is written f' or $\frac{df}{dx}$. The derivative function is generic, but then we have to specify at which point we are querying the derivative, since it only gives local information. For example, $f(x) = x^2$ has derivative $f'(x) = 2x$. By querying it at $x = 3$, we find that the derivative is $f'(3) = 6$, meaning that the function is increasing—because $f'(3)$ is positive—and doing so quite rapidly—the magnitude of 6 is quite high. Namely, it is increasing as we increase x. If instead we queried the derivative at $x = -0.2$, then we would get $f'(-0.2) = -0.4$, meaning locally the function is decreasing as we increase x, and it is doing so more slowly. Near zero, the derivative gets close to zero, because the function is flatter near that point and changing much more slowly. Note that we write $\frac{df}{dx}(3)$ rather than $\frac{df(3)}{dx}$, because this emphasizes that we take the derivative function $f' = \frac{df}{dx}$, and the evaluate it at 3. The derivative at two different points for this function is depicted in Figure 1.3b.

Useful derivative rules to remember include

$$
\begin{aligned}
&f(x) = x^a, && \frac{df}{dx}(x) = ax^{a-1} \\
&f(x) = \exp(x), && \frac{df}{dx}(x) = \exp(x) \\
&f(x) = \ln(x), && \frac{df}{dx}(x) = \frac{1}{x} \\
&f(x) = g(h(x)), \text{ set } u = h(x), && \frac{df}{dx}(x) = \left(\frac{df}{du}\frac{du}{dx}\right)(x) \qquad \triangleright \text{ chain rule for functions } g, h
\end{aligned}
$$

For example, for $f(x) = \exp(x^2)$, we use the chain rule with $u = x^2$ to get $f'(x) = \frac{d\exp}{du}\frac{du}{dx}(x) = \exp(u)2x = 2\exp(x^2)x$.

Exercise 5: What is the derivative of $f(x) = \exp(-x^3)$? □

1.3 Structure of the Book

The first part of the notes (Chapters 1-6) introduces you to probability, reasoning about uncertainty, modeling distributions and basic optimization concepts. The primary purpose of these chapters is to provide you with the needed mathematical background for the second half of the notes, focused more specifically on learning functions that make predictions (Chapter 7-12). The first half can feel more like a course in statistics, until we finally reach what feels more like a machine learning course in Chapter 7. Becoming better modelers is an important part of machine learning, and so Chapters 1-6 are not simply a course in statistics; this book attempts to make more explicit connections to machine learning along the way to emphasize this. Nonetheless, it is good to recognize that the primary role of Chapters 1-6 is to fill in the needed background, and that examples or topics may not always look directly connected to machine learning.

By the time we get to Chapter 7, the goal is to have a basic grasp of probability, optimization, how we formalize a modeling problem to learn the parameters of a distribution and how to reason about uncertainty in our estimates. The goal is also that you have gotten a bit more comfortable with the language of mathematics. Then we start to ask the questions posed at the start of this chapter: how do we formalize learning a function that makes predictions, how can we assess whether it is good—both conceptually and empirically—and how do we reason about uncertainty in our predictions.

Throughout the notes we revisit the same concepts, in ever increasing complexity. We first start with the simplest estimator: a sample average (or sample mean). For even this simple estimator, we can already ask questions about how we can be confident in our estimate, how many samples we need to get a good estimate and how we can modify our estimator to require fewer samples. Then we move on to reasoning about the parameters for simple distributions, and again revisit these questions. And finally we move on to more complex estimators, namely functions that make predictions for the two key settings in machine learning: regression and classification.

Chapter 2

Introduction to Probabilistic Modeling

Modeling the world around us and making predictions about the occurrence of events is a multidisciplinary endeavor standing on the solid foundations of probability theory, statistics, and computer science. Although intertwined in the process of modeling, these fields have relatively discernible roles and can be, to a degree, studied individually. Probability theory brings the mathematical infrastructure, firmly grounded in its axioms, for manipulating probabilities and equips us with a broad range of models with well-understood theoretical properties. Statistics contributes frameworks to formulate inference and the process of narrowing down the model space based on the observed data and our experience in order to find, and then analyze, solutions. Computer science provides us with theories, algorithms, and software to manage the data, compute the solutions, and study the relationship between solutions and available resources (time, space, computer architecture, etc.). As such, these three disciplines form the core quantitative framework for all of empirical science and beyond.

Probability theory and statistics have a relatively long history; the formal origins of both can be traced to the 17^{th} century. Probability theory was developed out of efforts to understand games of chance and gambling. The correspondence between Blaise Pascal and Pierre de Fermat in 1654 serves as the oldest record of modern probability theory. Statistics, on the other hand, originated from data collection initiatives and attempts to understand trends in the society (e.g., manufacturing, mortality causes, value of land) and political affairs (e.g., public revenues, taxation, armies). The two disciplines started to merge in the 18^{th} century with the use of data for inferential purposes in astronomy, geography, and social sciences. The increased complexity of models and availability of data in the 19^{th} century emphasized the importance of computing machines. This contributed to establishing the foundations of the field of computer science in the 20^{th} century, which is generally attributed to the introduction of the von Neumann architecture and formalization of the concept of an algorithm. The convergence of the three disciplines has now reached the status of a principled theory of probabilistic inference with widespread applications in science, business, medicine, military, political campaigns, etc. Interestingly, various other disciplines have also contributed to the core of probabilistic modeling. Concepts such as a Boltzmann distribution, a genetic algorithm, or a neural network illustrate the influence of physics, biology, psychology, and engineering.

We will refer to the process of modeling, inference, and decision making based on probabilistic models as *probabilistic reasoning* or reasoning under uncertainty. Some form of reasoning under uncertainty is a necessary component of everyday life. When driving, for example, we often make decisions based on our expectations about which way would be best to take. While these situations do not usually involve an explicit use of probabilities and probabilistic models, an intelligent driverless car such as Google Chauffeur must make

use of them. And so must a spam detection software in an email client, a credit card fraud detection system, or an algorithm that infers whether a particular genetic mutation will result in disease. Therefore, we first need to understand the concept of probability and then introduce a formal theory to incorporate evidence (e.g., data collected from instruments) in order to make good decisions in a range of situations.

At a basic level, probabilities are used to quantify the chance of the occurrence of events. As Jacob Bernoulli brilliantly put it in his work *The Art of Conjecturing* (1713), "To make a conjecture [prediction] about something is the same as to measure its probability. Therefore, we define the art of conjecturing [science of prediction] or stochastics, as the art of measuring probabilities of things as accurately as possible, to the end that, in judgements and actions, we may always choose or follow that which has been found to be better, more satisfactory, safer, or more carefully considered." The techniques of probabilistic modeling formalize many intuitive concepts. In a nutshell, they provide toolkits for rigorous mathematical analysis and inference, often in the presence of evidence, about events influenced by factors that we either do not fully understand or have no control of.

To provide a quick insight into the concept of uncertainty and modeling, consider rolling a fair six-sided die. We could accurately predict, or so we think, the outcome of a roll if we carefully incorporated the initial position, force, friction, shape defects, and other physical factors and then executed the experiment. But the physical laws may not be known, they can be difficult to incorporate or such actions may not even be allowed by the rules of the experiment. Thus, it is practically useful to simply assume that each outcome is equally likely; in fact, if we rolled the die many times, we would indeed observe that each number is observed roughly equally. Assigning an equal chance (probability) to each outcome of the roll of a die provides an efficient and elegant way of modeling uncertainties inherent to the experiment.

Another, more realistic example in which collecting data provides a basis for simple probabilistic modeling is a situation of driving to work every day and predicting how long it will take us to reach the destination tomorrow. If we recorded the "time to work" for a few months we would observe that trips generally took different times depending on many internal (e.g., preferred speed for the day) and also external factors (e.g., weather, road works, encountering a slow driver). While these events, if known, could be used to predict the exact duration of the commute, it is unrealistic to expect to have full information—rather we have *partial observability*. It is useful to provide ways of aggregating external factors via collecting data over a period of time and providing the distribution of the commute time. Such a distribution, in the absence of any other information, would then facilitate reasoning about events such as making it on time to an important meeting at 9 am.

The techniques of probabilistic modeling provide a formalism for dealing with such repetitive experiments influenced by a number of external factors over which we have little control or knowledge. With such a formalism, we can better understand and improve how we make predictions, because we can more clearly specify our assumptions about our uncertainty and explicitly reason about possible outcomes. In this chapter, we introduce probabilities and probability theory, from the beginning. Because probability is such a fundamental concept in machine learning, it is worth understanding where it comes from. Nonetheless, following the spirit of these notes, the treatment will be brief and focus mostly on what is needed to understand the development in following chapters.

2.1 Probability Theory and Random Variables

Probability theory is as a branch of mathematics that deals with measuring the likelihood of events. At the heart of probability theory is the concept of an *experiment*. An experiment can be the process of tossing a coin, rolling a die, checking the temperature tomorrow or figuring out the location of one's keys. When carried out, each experiment has an *outcome*, which is an element drawn from a set of predefined options, potentially infinite in size. The outcome of a roll of a die is a number between one and six; the temperature tomorrow might be a real number; the outcome of the location of one's keys can be a discrete set of places such as a kitchen table, under a couch, in office etc. In many ways, the main goal of probabilistic modeling is to formulate a particular question or a hypothesis pertaining to the physical world as an experiment, collect the data, and then construct a model. Once a model is created, we can compute quantitative measures of sets of outcomes we are interested in and assess the confidence we should have in these measures.

We can build up rules of probability, based on an elegantly simple set of axioms called the *axioms of probability*. Let the *sample space* (Ω) be a non-empty set of outcomes and the *event space* ($\mathcal{E}$) be a non-empty set of subsets of Ω. For example, $\Omega = \{1, 2, 3\}$ and one possible *event* is $A = \{1, 3\} \in \mathcal{E}$, where the event is that a 1 or a 3 is observed. Another possible event is $A = \{2\}$, where the event is that a 2 is observed. The **event space** $\mathcal{E}$ must satisfy the following properties[1]

1. $A \in \mathcal{E} \quad \Rightarrow \quad A^c \in \mathcal{E}$ (where A^c is the complement of the event A: $A^c = \Omega \backslash A$)
2. $A_1, A_2, \ldots \in \mathcal{E} \quad \Rightarrow \quad \bigcup_{i=1}^{\infty} A_i \in \mathcal{E}$
3. $\mathcal{E}$ is non-empty.

If $\mathcal{E}$ satisfies these three properties, then $(\Omega, \mathcal{E})$ is said to be a *measurable space*. The symbol $\varnothing$ means the empty set. Note that these three conditions imply that $\varnothing \in \mathcal{E}$ and $\Omega \in \mathcal{E}$.[2]

Now we can define the axioms of probability, which make it more clear why these two conditions are needed for our event space to define meaningful probabilities over events. A function $P : \mathcal{E} \to [0, 1]$ satisfies the **axioms of probability** if

1. $P(\Omega) = 1$
2. $A_1, A_2, \ldots \in \mathcal{E},\ A_i \cap A_j = \varnothing\ \forall i, j \quad \Rightarrow \quad P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$

is called a *probability measure* or a *probability distribution*. The tuple $(\Omega, \mathcal{E}, P)$ is called the *probability space*.

The second condition means that the probability of the union of disjoint sets equals the sum of their probabilities. This is an intuitive requirement. In it simplest form, this requirement states that if two events A_1, A_2 are disjoint, then $P(A_1 \cup A_2) = P(A_1) + P(A_2)$: the probability of either event occurring is the sum of their probabilities, because there is

[1] Such a set is usually called a *sigma algebra* or *sigma field*. This terminology feels daunting and is only due to historical naming conventions. Because the sigma algebra is simply the set of events to which we can assign probabilities (measure), we will use this more clear name. If you would like to learn more about this topic, it is more formally discussed in a branch of mathematics called Measure Theory.

[2] In fact, it is common to use the condition that $\Omega \in \mathcal{E}$ rather than using the condition that $\mathcal{E}$ is non-empty. However, it is equivalent and requiring that we have a non-empty event space is a more obvious condition to include.

no overlap in the outcomes in the events. More generally, a finite union should also satisfy this equality, namely that for any $A_1, A_2, \ldots A_N \in \mathcal{E}$, $A_i \cap A_j = \varnothing \ \forall i, j \Rightarrow P(\cup_{i=1}^N A_i) = \sum_{i=1}^N P(A_i)$. The above condition requires this equality for the union of infinitely many events, and implies that P satisfies this condition for finitely many sets. This is because all A_{N+j} for $j \geq 1$ can be set to the empty set $\varnothing$, where $A_i \cap \varnothing = \varnothing$ and $\varnothing \cap \varnothing = \varnothing$.

The beauty of these axioms lies in their compactness and elegance. Many useful expressions can be derived from the axioms of probability. For example, it is straightforward to show that $P(A^c) = 1 - P(A)$. This makes it more clear why we required that if an event is in the event space, then its complement should also be in the event space: if we can measure the probability of an event, then we know that the probability of that event not occurring is 1 minus that probability. Another property we can infer is that we always have $\Omega, \varnothing \in \mathcal{E}$, where $\varnothing$ corresponds to the event where nothing occurs—which must have zero probability.

Exercise 6: Show that for any event $A \in \mathcal{E}$, $P(A^c) = 1 - P(A)$. Assume that P satisfies the axioms of probability. □

Exercise 7: Show that the above three conditions on $\mathcal{E}$ imply that $\varnothing \in \mathcal{E}$ and $\Omega \in \mathcal{E}$. □

Example 1: [Discrete variables (countable)] Consider modeling the probabilities of the roll of a dice. The outcome space is the finite set $\Omega = \{1, 2, 3, 4, 5, 6\}$ and the event space $\mathcal{E}$ is the power set $\mathcal{P}(\Omega) = \{\varnothing, \{1\}, \{2\}, \ldots, \{2, 3, 4, 5, 6\}, \Omega\}$, which consists of all possible subsets of Ω. A natural probability distribution on $(\Omega, \mathcal{E})$ gives each dice roll a $1/6$ chance of occurring, defined as $P(\{x\}) = 1/6$ for $x \in \Omega$, $P(\{1, 2\}) = 1/3$ and so on. □

Example 2: [Continuous variables (uncountable)] Consider modeling the probabilities of the stopping time of a car, in the range of 3 seconds to 6 seconds. The outcome space is the continuous interval $\Omega = [3, 6]$. An event could be that the car stops within 3 to 3.1 seconds, giving $A = [3, 3.1] \in \mathcal{E}$. The probability $P(A)$ of such an event is likely low, because it would be a very fast stopping time. We could then start considering all possible time intervals for the event space, and corresponding probabilities. We can already see that this will be a bit more complicated for continuous variables, and so we more rigorously show how to define $\mathcal{E}$ and P below in Section 2.2.2. □

These two examples demonstrate the two most common cases we will encounter: discrete variables and continuous variables. The terms above—countable and uncountable—indicate whether a set can be enumerated or not. For example, you can iterate over the set of natural numbers, and so it is countable. The set of real numbers cannot be enumerated: you cannot provide a procedure to iterate over the reals, and so they are uncountable. Though this distinction results in real differences—such as using sums for countable sets and integrals for uncountable sets—the formalism and intuition will largely transfer between the two settings. We will focus on discrete and continuous variables. Much of the same ideas also transfer to mixed variables, where outcome spaces are composed of both discrete and continuous sets such as $\Omega = [0, 1] \cup \{2\}$. Further, for the uncountable setting, we specifically discuss continuous sets, i.e., those are unions of continuous intervals such as $\Omega = [0, 1] \cup [5, 10]$. Because almost all uncountable sets that we will want to consider are continuous, we will interchangeably use the terms continuous and uncountable to designate such spaces. Finally, discrete sets can either be finite, such as $\{1, 2, 3\}$, or countably infinite, such as the natural numbers. Continuous sets are clearly infinite, and are said to be uncountably infinite.

Before going further in-depth on how to define probability distributions, we first introduce *random variables*, and from here on will deal strictly with random variables. A random variable lets us more rigorously define transformations of probability spaces; once we execute that transformation, we can forget about the underlying probability space and can focus on the events and distribution only on the random variable. This is in fact what you do naturally when defining probabilities over variables, without needing to formalize it mathematically. Of course, here we will formalize it.

Consider again the dice example, where now instead you might want to know: what is the probability of seeing a low number (1-3) or a high number (4-6)? We can define a new probability space with $\mathcal{X} = \{\text{low}, \text{high}\}$, $\mathcal{E}_X = \{\varnothing, \{\text{low}\}, \{\text{high}\}, \mathcal{X}\}$ and $P_X(\{\text{low}\}) = 1/2 = P_X(\{\text{high}\})$. The transformation function $X : \Omega \to \mathcal{X}$ is defined as

$$X(\omega) \stackrel{\text{def}}{=} \begin{cases} \text{low} & \text{if } \omega \in \{1, 2, 3\} \\ \text{high} & \text{if } \omega \in \{4, 5, 6\} \end{cases}$$

The distribution P_X is immediately determined from this transformation. For example, $P_X(\{\text{low}\}) = P(\{\omega : X(\omega) = \text{low}\})$,[3] because the underlying probability space indicates the likelihood of seeing a $1, 2$ or 3. Now we can answer questions about the probability of seeing a low number or a higher number.

This function X is called a random variable.[4] The utility of this terminology is that we move from talking about probabilities of sets—which can be cumbersome—to writing boolean statements. For example, we write $P_X(X = x)$ or $P_X(X \in A)$, rather than $P(\{\omega : X(\omega) = x\})$ or $P(\{\omega : X(\omega) \in A\})$. For correctness, we can remember that it is a function defined on a more complex underlying probability space. But, in practice, we can start thinking directly in terms of the random variable X and the associated probabilities. Similarly, even for the dice role, we can acknowledge that there is a more complex underlying probability space, defined by the dynamics of the dice. When considering only the probabilities of discrete outcomes from 1-6, we have already implicitly applied a transformation on top of probabilities of the physical system.

Once we have a random variable, it defines a valid probability space $(\mathcal{X}, \mathcal{E}_X, P_X)$. Therefore, all the same rules of probability apply, the same understanding of how to define distributions, etc. In fact, we can always define a random variable X that corresponds to no transformation, to obtain the original probability space. For this reason, we can move forward assuming we are always dealing with random variables, without losing any generality. We will drop the subscripts, and consider $(\mathcal{X}, \mathcal{E}, P)$ to be defined for X.

2.2 Defining Distributions

Now we would like to know how to specify P to satisfy the axioms of probability, to model the probability of X taking values in an event A, $P(X \in A)$ with outcome space $\mathcal{X}$. This task feels daunting, because it seems we need to define the likelihood for every possible

[3] You can read $\{\omega : X(\omega) = \text{low}\}$ as "The event where the outcome was low" and more explicitly as "The set of outcomes where the outcome was low". You can read this whole expression $P(\{\omega : X(\omega) = \text{low}\})$ as "The probability of the event where the outcome was low".

[4] This terminology might be slightly confusing consider X is a function, so it is neither random, nor a variable. But, we end up using X as if its a variable that can take random outcomes, by writing statements like $X = x$. In this sense, the terminology is reasonable.

event—set of outcomes—and, in such a way that satisfies the axioms of probability, no less! Fortunately, instead we can define the distribution using a function defined directly on instances $x \in \mathcal{X}$. It is convenient to separately consider discrete (countable) and continuous (uncountable) sample spaces. For the discrete case, we will define probability mass functions and for the continuous case, we will define probability density functions.

2.2.1 Probability mass functions for discrete random variables

Let $\mathcal{X}$ be a discrete sample space and $\mathcal{E} = \mathcal{P}(\mathcal{X})$, the power set of $\mathcal{X}$. A function $p : \mathcal{X} \to [0, 1]$ is called a *probability mass function* (pmf) if

$$\sum_{x \in \mathcal{X}} p(x) = 1.$$

The probability of any event $A \in \mathcal{E}$ is defined as

$$P(A) \stackrel{\text{def}}{=} \sum_{x \in A} p(x).$$

It is straightforward to verify that P satisfies the axioms of probability and, thus, is a probability distribution. For discrete random variables, therefore, we will often write $P(X = x)$, which means that $P(X = x) = p(x)$ for each outcome $x \in \mathcal{X}$. We rarely, if ever, define the distribution directly, and rather define the pmf p which induces the distribution P.

Exercise 8: Show that the above P satisfies the axioms of probability. □

Example 3: Consider a roll of a fair six-sided die; i.e., $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$, and the event space $\mathcal{E} = \mathcal{P}(\mathcal{X})$. What is the probability that the outcome is a number greater than 4?

First, because the die is fair, we know that $p(x) = \frac{1}{6}$ for $\forall x \in \mathcal{X}$. Now, let A be an event in $\mathcal{E}$ that the outcome is greater than 4; i.e., $A = \{5, 6\}$. Thus,

$$P(A) = \sum_{x \in A} p(x) = \frac{1}{3}.$$

Notice that the distribution P is defined on the elements of $\mathcal{E}$, whereas p is defined on the elements of $\mathcal{X}$. That is, $P(\{1\}) = p(1)$, $P(\{2\}) = p(2)$, $P(\{1, 2\}) = p(1) + p(2)$, etc. □

To specify P, therefore, we need to determine how to specify the pmf, i.e., the probability of each discrete outcome. The pmf is often specified as a table of probability values. For example, to model the probability of a birthday for each day in the year, one could have a table of 365 values between zero and one, as long as the probabilities sum to 1. These probabilities could be computed from data about individuals birthdays, using counts for each day and normalizing by the total number of people in the population to estimate the probability of seeing a birthday on a given day. Such a table of values is very flexible, allowing precise probability values to be specified for each outcome. There are, however, a few useful named pmfs that have a (more restricted) functional form.

The *Bernoulli distribution* derives from the concept of a Bernoulli trial, an experiment that has two possible outcomes: success and failure. In a Bernoulli trial, a success occurs with probability $\alpha \in [0, 1]$ and, thus, failure occurs with probability $1 - \alpha$. A toss of a coin (heads/tails), a basketball game (win/loss), or a roll of a die (even/odd) can all be seen as

Bernoulli trials. The sample space consists of two elements and we define the probability of one of them as α. More specifically, $\mathcal{X} = \{\text{failure}, \text{success}\}$ and

$$p(x) = \begin{cases} \alpha & x = \text{success} \\ 1 - \alpha & x = \text{failure} \end{cases}$$

where $\alpha \in [0, 1]$ is a parameter. If we take instead that $\mathcal{X} = \{0, 1\}$, we can compactly write the Bernoulli distribution as

$$p(x) = \alpha^x (1 - \alpha)^{1-x} \tag{2.1}$$

for $x \in \mathcal{X}$. The Bernoulli distribution is often written Bernoulli(α). As we will see, a common setting where we use the Bernoulli is for binary classification, say where we try to predict whether a patient has the flu (outcome 0) or does not have the flu (outcome 1).

The *uniform distribution* for discrete sample spaces is defined over a finite set of outcomes each of which is equally likely to occur. Let $\mathcal{X} = \{1, \ldots, n\}$; then for $\forall x \in \mathcal{X}$

$$p(x) = \frac{1}{n}.$$

The uniform distribution does not contain parameters; it is defined by the size of the sample space. We refer to this distribution as Uniform(n). We will see later that the uniform distribution can also be defined over finite intervals in continuous spaces.

The *Poisson distribution* reflects the probability of how many incidences occur (implicitly within a fixed time interval). For example, a call center is likely to receive 50 calls per hour, with a much smaller probability of only receiving 5 calls or receiving as many as 1000 calls. This can be modeled with a Poisson(λ), where λ represents the expected number of calls. More formally, $\mathcal{X} = \{0, 1, \ldots\}$ and for $\forall x \in \mathcal{X}$

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}.$$

This mass function is hill-shaped, where the top of the hill is mostly centered around λ and there is a skew to having a short, steep left side of the hill and a long, less-steep right tail to the hill. The Poisson distribution is defined over an infinite sample space, but still countable. This is depicted in Figure 2.1.

Exercise 9: Prove that $\sum_{x=0}^{\infty} p(x) = 1$ for the Poisson distribution. Hint: use the power series expansion for the exponential, i.e., $\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. □

Example 4: As a prelude to estimating parameters to distributions, consider an example of how we might use a Bernoulli distribution and determine the parameter α to the Bernoulli. A canonical example for Bernoulli distributions is a coin flip, where the outcomes are heads (H) or tails (T). $P(X = \text{H}) = \alpha$ is the probability of seeing H and $P(X = \text{T}) = 1 - \alpha$ is the probability of seeing T. We commonly assume $\alpha = 0.5$; this is called a fair (unbiased) coin. If we flipped the coin many times, we would expect to see about the same number of H and T. However, a *biased* coin may have some skew towards H or T. If we flipped the coin many times, if $\alpha > 0.5$ we should eventually notice more H come up and if $\alpha < 0.5$, we should notice more T.

How might we actually determine this α? An intuitive idea is to use repeated experiments (data), just as described above: flip the coin many times to see if you can gauge the

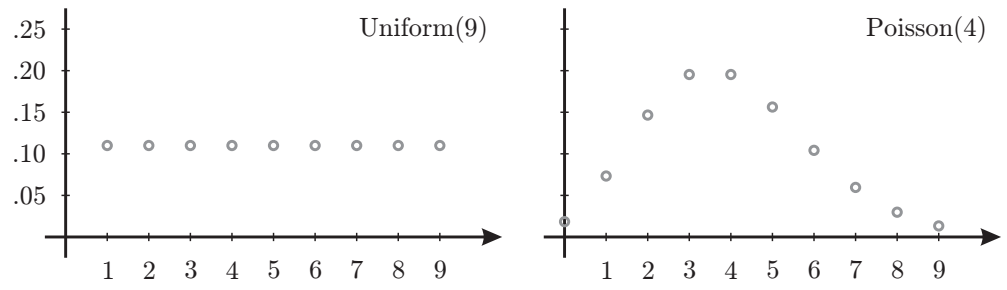


Figure 2.1: Two probability mass functions, for discrete random variables. The Poisson distribution continues further on the x-axis (for variable $x \in \mathbb{N}$), with probability decreasing to zero as $x \to \infty$.

skew. If you see 1000 H and 50 T, a natural guess for the bias is $\alpha = \frac{1000}{1000+50} \approx 0.95$. How confident are you in this solution? Is it definitely 0.95? And how do we more formally define why this should be the solution? This is in fact a reasonable solution, and corresponds to the maximum likelihood solution, as we will discuss in Chapter 5. □

2.2.2 Probability density functions for continuous random variables

The treatment of continuous probability spaces is analogous to that of discrete spaces, with probability density functions (pdfs) replacing probability mass functions and integrals replacing sums. In defining pdfs, however, we will not be able to use tables of values, and will be restricted to functional forms. The main reason for this difference stems from the fact that it no longer makes sense to measure the probability of a singleton event.

Consider again the stopping time for a car, discussed in Example 2. It would not make a lot of sense to ask the probability of the car stopping in exactly 3.14159625 seconds; realistically, the probability of such a precise event is vanishingly small. In fact, the probability of seeing precisely that stopping time is zero, because the set $\{3.14159625\}$ as a subset of $[3, 6]$ is a set of measure zero. Essentially, it takes up zero mass inside the interval $[3, 6]$, which is after all uncountably infinite. Instead, we will have to consider the probabilities of intervals, like $[4, 5]$ or $[5.667, 5.668]$.

For continuous spaces, we will assume that the set of events $\mathcal{E}$ consists of all possible intervals, called the Borel field $\mathcal{B}(\mathcal{X})$. For example, if $\mathcal{X} = \mathbb{R}$, the Borel field $\mathcal{B}(\mathbb{R})$ consists of all open intervals (e.g., $(0, 1)$), closed intervals (e.g.,$[0, 1]$) and semi-open intervals (e.g., $[0, 1)$) in $\mathbb{R}$, as well as sets that can be obtained by a countable number of basic set operations on them, such as unions. This results in a more restricted set of events than the power set of $\mathcal{X}$, which would, for example, include sets with only a singleton event. $\mathcal{B}(\mathbb{R})$ is still a huge set—an uncountably infinite set—but still smaller than $\mathcal{P}(\mathbb{R})$. Nicely, though, $\mathcal{B}(\mathbb{R})$ still contains all sets we could conceivably want to measure. The Borel field can be defined for any measurable space, such as higher-dimensional spaces like $\mathcal{X} = \mathbb{R}^2$, with events such as $A = [0, 1] \times [0, 1] \subset \mathcal{X}$ or $A = [1, 2] \times [-1, 4] \cup [0, 0.1] \times [10, 1000]$.

Notational Remark: $[0, 1] \times [0, 1]$ is the 2-dimensional space consisting of all pairs (x, y) where $x \in [0, 1]$ and $y \in [0, 1]$. The union of $[1, 2] \times [-1, 4]$ and $[0, 0.1] \times [10, 1000]$ consists of all pairs (x, y) where $x \in [1, 2] \cup [0, 0.1]$ and $y \in [-1, 4] \cup [10, 1000]$.

Let $\mathcal{X}$ be a continuous sample space and $\mathcal{E} = \mathcal{B}(\mathcal{X})$. A function $p : \mathcal{X} \to [0, \infty)$ is called a *probability density function* (pdf) if[5]

$$\int_{\mathcal{X}} p(x)dx = 1.$$

The probability of an event $A \in \mathcal{B}(\mathcal{X})$ is defined as

$$P(A) \stackrel{\text{def}}{=} \int_A p(x)dx.$$

Notice that the definition of the pdf is not restricted to having a range $[0, 1]$, but rather to $[0, \infty)$. For pmfs, the probability of a singleton event $\{x\}$ is the value of the pmf at the sample point x; i.e., $P(\{x\}) = p(x)$. Since probability distributions P are restricted to the range $[0, 1]$, this implies pmfs must also be restricted to that range. In contrast, the value of a pdf at point x is not a probability; it can actually be greater than 1. Though it is common when speaking informally to call $p(x)$ the probability of x, more accurately we call this the density at x because it is most definitely not a probability. In fact, as mentioned above, the probability at any single point is 0 (i.e., a countable subset of $\mathcal{X}$ is a set of measure zero).

A natural confusion is how p can integrate to 1, but actually have values larger than 1. The reason for this is that $p(x)$ can be (much) larger than 1, as long as its only for a very small interval. Consider the small interval $A = [x, x + \Delta x]$, with probability

$$P(A) = \int_x^{x+\Delta x} p(x)dx \approx p(x)\Delta x.$$

A potentially large value of the density function is compensated for by the small interval Δx to result in a number between 0 and 1. So, even if $p(x)$ is a million, and the density of points in the small interval or ball around x is large, the probability of an event must still be ≤ 1. The density does indicate that there is high likelihood around that point. By having a huge density around x, this suggests that the density for other points is zero or near zero and that the pdf is extremely peaked around x.

Unlike pmfs, we cannot so easily define pdfs p to flexibly provide specific probabilities for each outcome with a table of probabilities. Rather, for pdfs, we will usually use a known pdf that satisfies the required properties. Further, unlike the discrete case, we will never write $P(X = x)$, because that would be zero. Rather, we will typically write $P(X \in A)$, such as by writing $P(X \in [0.1, 0.2])$ or $P(X \leq 5)$. We highlight four pdfs here, that will be used throughout this book.

The *uniform distribution* is defined by an equal value of a probability density function over a finite interval in $\mathbb{R}$. Thus, for $\mathcal{X} = [a, b]$ the uniform probability density function $\forall x \in [a, b]$ is defined as

$$p(x) \stackrel{\text{def}}{=} \frac{1}{b - a}.$$

[5]For correctness, we would like to note that this definition uses Lebesgue integration. You do not need to know about nuanced differences in integration formulations; for all settings we consider, all the definitions are equivalent and your knowledge of integration rules will be effective.

[6]Some images in this document are taken from other sources. This is not good practice, and these images will be replaced someday soon. We want to highlight that we do not encourage this for formal documents, but only use them here in these educational notes for your benefit temporarily.

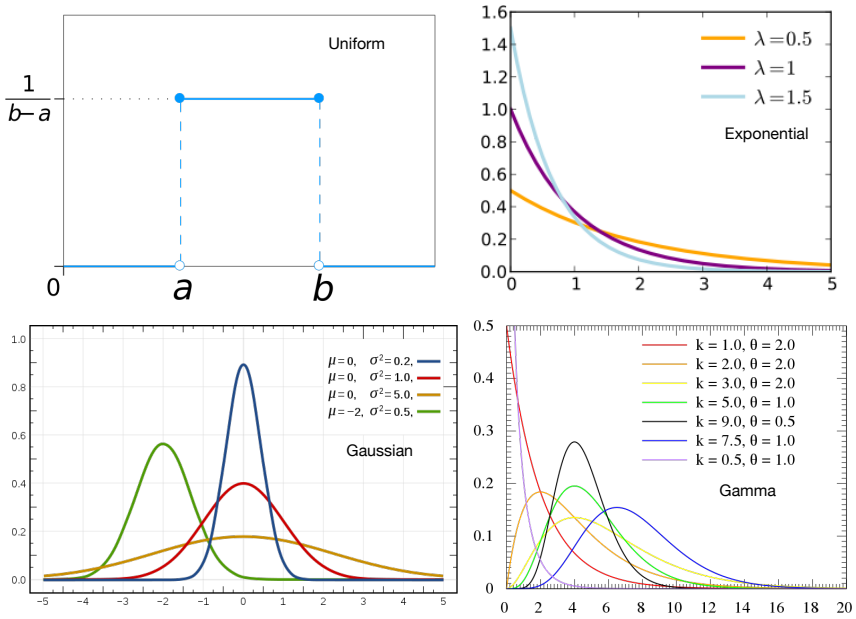


Figure 2.2: Four probability density functions, for continuous random variables. Images taken from Wikipedia.[6]

One can also define Uniform(a, b) by taking $\mathcal{X} = \mathbb{R}$ and setting $p(x) = 0$ whenever x is outside of $[a, b]$. This form is convenient because $\mathcal{X} = \mathbb{R}$ can then be used consistently for all one-dimensional probability distributions. When we do this, we will refer to the subset of $\mathbb{R}$ where $p(x) > 0$ as the *support* of the density function.

The *exponential distribution* is defined over a set of non-negative numbers; i.e., $\mathcal{X} = [0, \infty)$. For parameter $\lambda > 0$, its pdf is

$$p(x) = \lambda e^{-\lambda x}.$$

As the name suggests, this pdf has an exponential form, with sharply decreasing probability for values x as they increase in magnitude. As before, the sample space can be extended to all real numbers, in which case we would set $p(x) = 0$ for $x < 0$.

The *Gaussian distribution* or normal distribution is one of the most frequently used probability distributions. It is defined over $\mathcal{X} = \mathbb{R}$, with two parameters, $\mu \in \mathbb{R}$ and $\sigma > 0$ and pdf

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

As we will discuss next, for a random variable that is Gaussian distributed, the parameter μ is the mean or expected value and σ^2 is the variance. We will refer to this distribution as Gaussian(μ, σ^2) or $\mathcal{N}(\mu, \sigma^2)$. When the mean is zero, and the variance is 1 (unit variance), this Gaussian is called the *standard normal*. This specific Gaussian has a name because it is so frequently used. Both Gaussian and exponential distributions are members of a broader family of distributions called the *natural exponential family*.

Figure 2.3: Selection of a random number (x) from the unit interval $[0, 1]$.

The *Laplace distribution* is similar to the Gaussian, but is more peaked around the mean. It is also defined over $\mathcal{X} = \mathbb{R}$, with two parameters, $\mu \in \mathbb{R}$ and $b > 0$ and pdf

$$p(x) = \frac{1}{2b} e^{-\frac{1}{b}|x-\mu|}$$

The *gamma distribution* is used to model waiting times, and is similar to the Poisson distribution but for continuous variables. It is defined over $\mathcal{X} = (0, \infty)$, with shape parameter $\alpha > 0$ and rate parameter $\beta > 0$ and pdf

$$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

where $\Gamma(\alpha)$ is called the gamma function. A random variable that is gamma-distributed is denoted $X \sim \text{Gamma}(\alpha, \beta)$.

Example 5: Consider selecting a number (x) between 0 and 1 uniformly randomly (Figure 2.3). What is the probability that the number is greater than or equal to $\frac{3}{4}$ or less than and equal to $\frac{1}{4}$?

We know that $\mathcal{X} = [0, 1]$. The distribution is defined by the uniform pdf, $p(x) = \frac{1}{b-a} = 1$ where $a = 0, b = 1$ define the interval for the outcome space. We define the event of interest as $A = \left[0, \frac{1}{4}\right] \cup \left[\frac{3}{4}, 1\right]$ and calculate its probability as

$$\begin{aligned} P(A) &= \int_0^{1/4} p(x)dx + \int_{3/4}^{1} p(x)dx \qquad \triangleright\ p(x) = 1 \\ &= \left(\frac{1}{4} - 0\right) + \left(1 - \frac{3}{4}\right) \\ &= \frac{1}{2}. \end{aligned}$$

What if we had instead asked the probability that the number is *strictly greater* than $\frac{3}{4}$ or *strictly less* than $\frac{1}{4}$? Because the probability of any individual event in the continuous case is 0, there is no difference in integration if we consider open or closed intervals. Therefore, the probability would still be $\frac{1}{2}$. □

Example 6: Let's imagine you have collected your commute times for the year[7], and would like to model the probability of your commute time to help you make predictions about your commute time tomorrow. For this setting, your random variable X corresponds to the commute time, and you need to define probabilities for this random variable. This data could be considered to be discrete, taking values, in minutes, $\{4, 5, 6, \ldots, 26\}$. You could then create histograms of this data (table of probability values), as shown in Figure 2.4, to reflect the likelihood of commute times.

[7] as my amazing colleague Predrag amazingly did, and you get to see his fascinating data here.

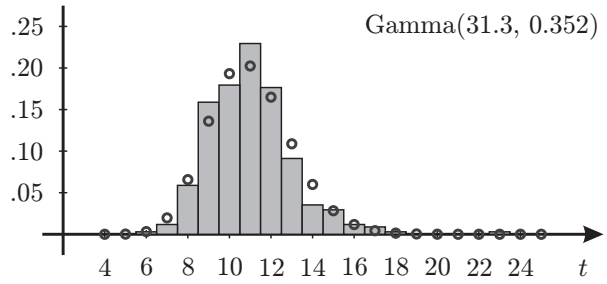


Figure 2.4: A histogram of recordings of the commute time (in minutes) to work. The data set contains 340 measurements collected over one year, for a distance of roughly 3.1 miles. The data was modeled using a gamma family of probability distributions, with the particular location and scale parameters estimated from the raw data. The values of the gamma distribution are shown as dark circles.

The commute time, however, is not actually discrete, and so you would like to model it as a continuous RV. One reasonable choice is a gamma distribution. How, though, does one take the recorded data and determine the parameters α, β to the gamma distribution? Estimating these parameters is actually quite straightforward, though not as immediately obvious as estimating tables of probability values; we discuss how to do so in Chapter 5. The learned gamma distribution is also depicted in Figure 2.4.

Given the gamma distribution, one could now ask the question: what is the most likely commute time today? This corresponds to $\max_x p(x)$, which is called the *mode* of the distribution. Another natural question is the average or expected commute time. To obtain this, you need the expected value (mean) of this gamma distribution, which we define below in Section 2.4. □

2.3 Multivariate Random Variables

Much of the above development extends to multivariate random variables—a vector of random variables—because the definition of outcome spaces and probabilities is general. The examples so far, however, have dealt with scalar random variables, because for multivariate random variables, we need to understand how variables interact. In this section, we discuss several new notions that only arise when there are multiple random variables, including joint distributions, conditional distributions, marginals and dependence between variables.

Let us start with a simpler example, with two discrete random variables X and Y with outcome spaces $\mathcal{X}$ and $\mathcal{Y}$. There is a joint probability mass function $p : \mathcal{X} \times \mathcal{Y} \to [0, 1]$, and corresponding joint probability distribution P, such that

$$p(x, y) \stackrel{\text{def}}{=} P(X = x, Y = y)$$

where the pmf needs to satisfy

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) = 1.$$

This fits within the definition of probability spaces, because $\Omega = \mathcal{X} \times \mathcal{Y}$ is a valid space, and $\sum_{\omega \in \Omega} p(\omega) = \sum_{(x,y) \in \Omega} p(x, y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y)$. The random variable $Z = (X, Y)$

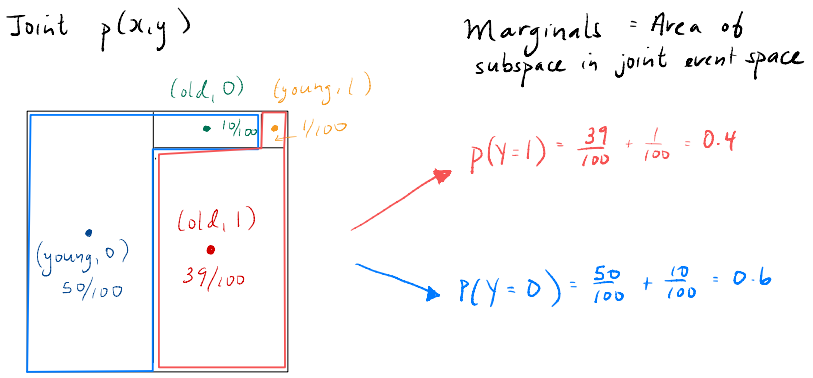


Figure 2.5: The joint distribution and two marginal distributions, for the two RVs X, Y where $x \in \{young, old\}$ and $y \in \{0, 1\}$ where 0 means No Arthritis and 1 means Arthritis. The marginal distribution $p(y)$ considers the probability of seeing $Y = 0$, with either $X = young$ or $X = old$, and $Y = 1$, with either $X = young$ or $X = old$. The area of the subspace where $y = 0$ gives us $p(Y = 0$, and the (remaining) area where $y = 1$ gives us $p(Y = 1)$.

is a multivariate random variable, of two dimensions. For example, if $\mathcal{X} = \{\text{young}, \text{old}\}$ and $\mathcal{Y} = \{\text{no arthritis}, \text{arthritis}\}$, then the pmf could be the table of joint[8] probabilities.

		Y	
		no arthritis	1
X	young	$P(X{=}\text{young}, Y{=}\text{no arthritis}) = 1/2$	$P(X{=}\text{young}, Y{=}\text{arthritis}) = 1/100$
	old	$P(X{=}\text{old}, Y{=}\text{no arthritis}) = 1/10$	$P(X{=}\text{old}, Y{=}\text{arthritis}) = 39/100$

Table 2.1: A joint probability table for random variables X and Y.

We can see that the two random variables interact, by looking at the joint probabilities in the table. For example, the joint probability is small for young and having arthritis. This distribution is over both variables, and so the probabilities in the whole table must sum to 1. We can see that they do, since 1/2+1/100+1/10+39/100 = 1. Notice, however, that the rows do not sum to 1 nor do the columns.

Given a joint distribution over random variables, one would hope that we could extract more specific probabilities, like the distribution over just one of those variables, which is called the *marginal distribution*. The marginal can be simply computed, by summing up over all values of the other variable, visualized in Figure 2.5

$$P(X = \text{young}) = p(\text{young}, \text{no arthritis}) + p(\text{young}, \text{arthritis}) = \tfrac{51}{100}.$$

A young person either does or does not have arthritis, so summing up over these two possible cases factors out that variable. Therefore, using data collected for random variable $Z = (X, Y)$, we can determine the proportion of the population that is young and the proportion that is old. Notice that we do not need to marginalize to get the proportion that are old, instead we can use $P(X = \text{old}) = 1 - P(X = \text{young}) = 49/100$.

[8] Pun intended.

In general, we can consider d-dimensional random variable $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$ with vector-valued outcomes $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, such that each x_i is chosen from some $\mathcal{X}_i$. Then, for the discrete case, any function $p : \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_d \to [0, 1]$ is called a multidimensional probability mass function if

$$\sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \cdots \sum_{x_d \in \mathcal{X}d} p\left(x_1, x_2, \ldots, x_d\right) = 1.$$

or, for the continuous case, $p : \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_d \to [0, \infty]$ is a multidimensional probability density function if

$$\int_{\mathcal{X}_1} \int_{\mathcal{X}_2} \cdots \int_{\mathcal{X}d} p\left(x_1, x_2, \ldots, x_d\right) dx_1 dx_2 \ldots dx_d = 1.$$

A *marginal distribution* is defined for a subset of $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$ by summing or integrating over the remaining variables. For the discrete case, the marginal distribution $p\left(x_i\right)$ is defined as

$$p\left(x_i\right) \stackrel{\text{def}}{=} \sum_{x_1 \in \mathcal{X}_1} \cdots \sum_{x_{i-1} \in \mathcal{X}_{i-1}} \sum_{x_{i+1} \in \mathcal{X}_{i+1}} \cdots \sum_{x_d \in \mathcal{X}_d} p\left(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_d\right),$$

where the variable x_i is fixed to some value and we sum over all possible values of the other variables. Similarly, for the continuous case, the marginal distribution $p\left(x_i\right)$ is defined as

$$p\left(x_i\right) \stackrel{\text{def}}{=} \int_{\mathcal{X}_1} \cdots \int_{\mathcal{X}_{i-1}} \int_{\mathcal{X}_{i+1}} \cdots \int_{\mathcal{X}_d} p\left(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_d\right) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_d.$$

Notational Remark: Notice that we use p to define the density over $\boldsymbol{x}$, but then we overload this terminology and also use p for the density only over x_i. To be more precise, we should define two separate functions (pdfs), say $p_{\boldsymbol{x}}$ for the density over the multivariate random variable and p_{x_i} for the marginal. It is common, however, to simply use p, and infer the random variable from context. In most cases, it is clear; if it is not, we will explicitly highlight the pdfs with additional subscripts.

2.3.1 Conditional distributions

Conditional probabilities define probabilities of a random variable X, given information about the value of another random variable Y. More formally, the conditional probability $p(y|x)$ for two random variables X and Y is defined as

$$p(y|x) \stackrel{\text{def}}{=} \frac{p(x, y)}{p(x)} \tag{2.2}$$

where $p(x) > 0$. To be more precise, we have three distributions: the joint distribution $p_{x,y}(x, y)$, the marginal $p_x(x)$ and the conditional distribution $p_{y|x}(y|x)$. We can equivalently think of this conditional distribution as a distribution over y that is different for different x. For example, if $\mathcal{X} = \{0, 1\}$, then we have two different distribution $p_0(y)$ and $p_1(y)$, or as written above $p(y|X = 0)$ and $p(y|X = 1)$.

Exercise 10: Verify that $p(y|x)$ sums (integrates) to 1 over all values $y \in \mathcal{Y}$ for a fixed given $x \in \mathcal{X}$, and thus satisfies the conditions of a probability mass (density) function. $\square$

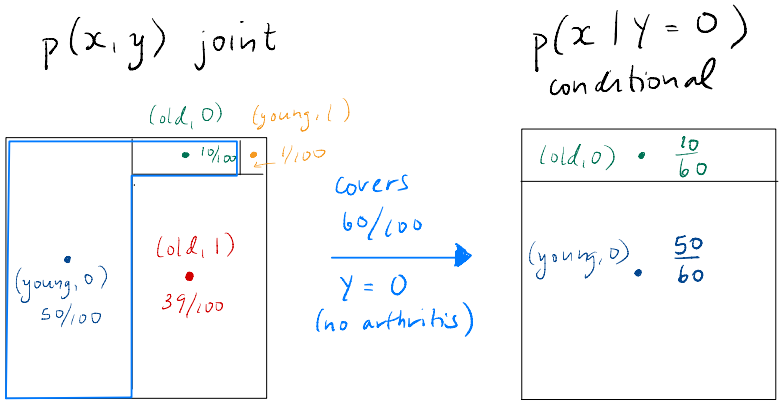


Figure 2.6: The joint distribution and a conditional distribution, for the two RVs X, Y where $x \in \{young, old\}$ and $y \in \{0, 1\}$ where 0 means No Arthritis and 1 means Arthritis. The conditional distribution looks at subspace in the event space, where $Y = 0$. Restricted to that subspace, we can reason about the relative probabilities of events, by looking at their relative areas. The relative probability can then be obtained by renormalizing by their combined area, which is precisely what normalizing by $p(Y = 0)$ does. Jointly, the events $(young, 0)$ and $(old, 0)$ cover 60% of the space (60/100). Normalizing by $p(Y = 0) = 60/100$ gives new probabilities $0.5/0.6 = 0.8\bar{3}$ and $0.5/0.6 = 0.1\bar{6}$.

Equation (2.2) now allows us to calculate the posterior probability of an event A, given some observation x, as

$$P(Y \in A|X = x) = \begin{cases} \sum_{y \in A} p(y|x) & Y : \text{discrete} \\ \\ \int_A p(y|x)dy & Y : \text{continuous} \end{cases}$$

Two important rules for conditional distributions are the *product rule* and *Bayes' rule*. The *product rule* states that

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \tag{2.3}$$

We can derive *Bayes' rule*, using the product rule:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{2.4}$$

Therefore, one really only needs to remember the product rule, to easily recall Bayes' rule.

Example 7: Let us return to the arthritis example. Using the product rule and Bayes rule, we can also answer questions about conditional distributions. Imagine we want to answer: what is the probability of having arthritis, if you are young? This equates to computing $P(Y = \text{arthritis}|X = \text{young})$. Recall that we showed

$$P(X = \text{young}) = p(\text{young}, \text{no arthritis}) + p(\text{young}, \text{arthritis}) = \tfrac{51}{100}.$$

Given this, and the fact that $p(y|x) = p(x, y)/p(x)$, we can compute

$$P(Y = \text{arthritis}|X = \text{young}) = p(\text{young}, \text{arthritis})/P(X = \text{young}) = (\tfrac{1}{100})/(\tfrac{51}{100}) = \tfrac{1}{51} \approx 0.0196.$$

We can also ask questions like: what is the probability of a person being young, given they have arthritis? This equates to computing $P(X = \text{young}|Y = \text{arthritis})$. We can use the same strategy as above to compute this. But, there are multiple ways to get this answer. This time, let's use Bayes rule and the above answer to compute this.

$$P(X = \text{young}|Y = \text{arthritis}) = P(Y = \text{arthritis}|X = \text{young})P(X = \text{young})/P(Y = \text{arthritis}).$$

To use this formula, we have to get $P(Y = \text{arthritis})$, which we know how to do using marginalization. The remaining steps are left as an exercise; see if you can get the right answer of 0.025. We additionally visualize $p(x|Y = \text{no arthritis})$ in Figure 2.6. □

The product rule can be extended to more than two variables. We can write

$$p(x_1, \ldots, x_d) = p(x_d|x_1, \ldots, x_{d-1})p(x_1, \ldots, x_{d-1}).$$

By a recursive application of the product rule, we obtain

$$\begin{aligned} p(x_1, \ldots, x_d) &= p(x_d|x_1, \ldots, x_{d-1})p(x_1, \ldots, x_{d-1}) \\ &= p(x_d|x_1, \ldots, x_{d-1})p(x_{d-1}|x_1, \ldots, x_{d-2})p(x_1, \ldots, x_{d-2}) \\ &\vdots \\ &= p(x_d|x_1, \ldots, x_{d-1})p(x_{d-1}|x_1, \ldots, x_{d-2}) \ldots p(x_2|x_1)p(x_1). \end{aligned}$$

More compactly,

$$p(x_1, \ldots, x_d) = p(x_1) \prod_{i=2}^{d} p(x_i|x_1, \ldots, x_{i-1}) \tag{2.5}$$

which is referred to as the *chain rule* or *general product rule*. For example, for three variables, the product rule gives

$$p(x_1, x_2, x_3) = p(x_3|x_2, x_1)p(x_2|x_1)p(x_1)$$

You may notice that the order of variables in the product rule did not seem to matter. It is in fact somewhat interesting that we can either define the conditional distribution $p(x|y)$ and marginal $p(y)$ or we can define $p(y|x)$ and $p(x)$ and both equivalently recover the joint distribution $p(x, y)$. This property is simply a fact of the definition of conditional distributions, and provides flexibility when estimating distributions. We will mostly use this equivalence in the form of Bayes rule, when doing parameter estimation and maximum likelihood. For work in graphical models, which is not discussed here, this flexibility is of even greater importance.

2.3.2 Independence of random variables

Two random variables are *independent* if their joint probability distribution factors into the product of the marginals

$$p(x, y) = p(x)p(y).$$

One intuitive reason for this definition can be seen by considering X conditioned on Y. If $p(x|y) = p(x)$, then this means that the value of Y has no influence on the distribution over X, and so they are independent. From the product rule, we know $p(x, y) = p(x|y)p(y)$ and since $p(x|y) = p(x)$, this gives $p(x, y) = p(x)p(y)$ as defined above.

The notion of independence can be generalized to more than two random variables. More generally, d random variables are said to be *mutually independent* or *jointly independent* if a joint probability distribution of any subset of variables can be expressed as a product of marginal probability distributions of its components

$$p\left(x_1, x_2, \ldots, x_d\right) = p(x_1)p(x_2)\ldots p(x_d).$$

Another form of independence, called *conditional independence*, is used even more frequently in machine learning. It represents independence between variables in the presence of some other random variable (evidence); e.g.,

$$p\left(x, y|z\right) = p(x|z)p(y|z)$$

Interestingly, the two forms of independence are unrelated: neither one implies the other. X and Y can be independent, but not conditionally independent given Z. X and Y can be conditionally independent given Z, but not independent. We expand on this in Example 8.

Independence and conditional independence are critical in machine learning. If two variables are independent, this has important modeling implications. For example, if feature X and target Y are independent, then X is not useful for predicting Y and so is not a useful feature. If two variables are conditionally independent given another variable, this can also have important modeling implications. For example, if we have two features X_1 and X_2, with target Y, where X_2 and Y are conditionally independent given X_1, then feature X_2 is redundant and could potentially be discarded.

As a concrete example, let X_1 = temperature in Celcius and X_2 = temperature in Fahrenheit, with Y = plants need watering. Y is definitely not independent of X_2; however, once X_1 is known (or given), then there is no additional information to be gained from X_2 and so $p(y|x_1, x_2) = p(y|x_1) = p(y|x_2)$. In general, recognizing independencies and conditional independencies can inform and simplify the modeling procedure. We end this section with one more example, using a biased coin, to highlight the distinction between independence and conditional independence.

Example 8: [Biased coin and conditional independence] Assume a manufacturer has produced a biased coin, where it does not equally randomly give heads (H) or tails (T). Rather, it actually has some unknown probability α of seeing H when flipping the coin. Because this bias is unknown, we will encode our uncertainty by defining a random Z = bias of the coin. In general, this random variable can take values in $[0, 1]$. For the purposes of this example, let's make this a bit simpler, and assume that we know the bias is one of $\mathcal{Z} = \{0.1, 0.5, 0.8\}$. If the bias is 0.5, that would mean this is an unbiased (fair) coin. Let's further assume that we think the probability of each bias is equally likely, meaning $P(Z = z) = 1/3$, because the manufacturer gave us no reason to think any of 0.1, 0.5 or 0.8 to be more likely.

Now imagine that you flip the coin twice, and record the two outcomes x_1 and x_2. These two separate flips correspond to two random variables, X_1 and X_2. The outcome space for X_1 and X_2 is $\{\mathrm{H}, \mathrm{T}\}$. Given the true bias of the coin, z, the distribution is Bernoulli $P(X_i = \mathrm{H}|Z = z) = z$, i.e., X_i is a Bernoulli random variable with parameter z. Because

we do not know the true bias, we have to marginalize over Z to get the marginal distribution over X_1,

$$\begin{aligned}
P(X_1 = x) &= \sum_{z \in \mathcal{Z}} P(X_1 = x, Z = z) \\
&= \sum_{z \in \mathcal{Z}} P(X_1 = x | Z = z) P(Z = z) && \triangleright \text{ product rule} \\
&= P(X_1 = x | Z = 0.1) P(Z = 0.1) + P(X_1 = x | Z = 0.5) P(Z = 0.5) + \\
&\quad + P(X_1 = x | Z = 0.8) P(Z = 0.8)
\end{aligned}$$

Therefore

$$P(X_1 = H) = 0.1\tfrac{1}{3} + 0.5\tfrac{1}{3} + 0.8\tfrac{1}{3} \approx 0.467$$

which does not correspond to the bias of any of the coins.

Now let us reason about independence and conditional independence. First, are X_1 and X_2 conditionally independent given Z? The answer is yes, because given the bias of the coin, knowing the outcome of X_2 does not influence the distribution over X_1, i.e.,

$$P(X_1 = x_1, X_2 = x_2 | Z = z) = P(X_1 = x_1 | Z = z) P(X_2 = x_2 | Z = z)$$

Regardless of what we observe for X_2, we know the distribution over X_1 is a Bernoulli with the given bias z. For example, imagine I hand you a coin and say: the bias of this coin is 0.8. Then, you know that the probability of seeing heads on the first flip is 0.8, and also 0.8 on the second flip regardless of the outcome of the first flip.

Are X_1 and X_2 independent? The answer is no, because without knowing the bias of the coin, knowing the outcome of X_2 tells us something about the Bernoulli distribution over X_1. For example, if $X_1 = \text{T}$ and $X_2 = \text{H}$, then the second outcome suggests that the bias might not be totally skewed towards T. More formally,

$$\begin{aligned}
P(X_1 = x_1, X_2 = x_2) &= \sum_{z \in \mathcal{Z}} P(X_1 = x_1, X_2 = x_2 | Z = z) P(Z = z) \\
&= \sum_{z \in \mathcal{Z}} P(X_1 = x_1 | Z = z) P(X_2 = x_2 | Z = z) P(Z = z)
\end{aligned}$$

which is not guaranteed to equal $P(X_1 = x_1) P(X_2 = x_2)$, where

$$\begin{aligned}
&P(X_1 = x_1) P(X_2 = x_2) \\
&= \left(\sum_{z_1 \in \mathcal{Z}} P(X_1 = x_1 | Z = z_1) P(Z = z_1) \right) \left(\sum_{z_2 \in \mathcal{Z}} P(X_2 = x_2 | Z = z_2) P(Z = z_2) \right)
\end{aligned}$$

□

The key point of this example is that the distribution reflects our beliefs and knowledge about the world, rather than corresponding to an objective truth. Because we do not know the bias of the coin, we write our distributions to reflect only what we know. The world, of course, knows the bias of the coin, and the outcome follows the rules of these true underlying probabilities. But our goal is to formalize and reason about what we know. In this sense, it is helpful to think of probabilities as our modeling tools, to express our beliefs about what we know. In the above, we were able to express what we know about the outcomes of the coin, by considering all possible biases and their probabilities.

2.4 Expectations and Moments

The *expected value*, or *mean*, of a random variable X is the average of repeatedly sampled x, in the limit of sampling. It is not necessarily the value we expect to see most frequently—that is called the mode. More precisely, given the pmf or pdf p for outcome space $\mathcal{X}$, the expectation of X is

$$\mathbb{E}[X] \stackrel{\text{def}}{=} \begin{cases} \sum_{x \in \mathcal{X}} xp(x) & \text{if } X \text{ is discrete} \\ \\ \int_{\mathcal{X}} xp(x)dx & \text{if } X \text{ is continuous} \end{cases}$$

For a dice roll, where each number from 1 to 6 has uniform probability, the expected value is 3.5 and the mode is tied for all numbers (i.e., it is multi-modal). For a Bernoulli distribution, where $\mathcal{X} = \{0, 1\}$, the expected value is α, which is not an outcome that will be observed, but is the average of 0s and 1s if we flipped the coin infinitely many times. The mode in this case depends on α: if $\alpha > 0.5$, making 1 have higher probability, then the mode is 1; if $\alpha < 0.5$, the mode is 0; otherwise, it is bimodal with modes 0 and 1. For a Gaussian distribution, the expected value is the parameter μ, and the mode also equals μ.

In general, we may be interested in the expected value of functions of the random variable X. For example, we may want to know $\mathbb{E}[X^2]$, or more generally $\mathbb{E}\left[X^k\right]$ for some $k > 1$. Or, we may want to know $\mathbb{E}\left[(X - c)^k\right]$ for some $k > 1$ and a constant c. These are called the *moments* of X. In general, for a function $f : \mathcal{X} \to \mathbb{R}$, we can consider $f(X)$ to be a transformed random variables and define its expectation as

$$\mathbb{E}[f(X)] = \begin{cases} \sum_{x \in \mathcal{X}} f(x)p(x) & \text{if } X \text{ is discrete} \\ \\ \int_{\mathcal{X}} f(x)p(x)dx & \text{if } X \text{ is continuous} \end{cases}$$

If $\mathbb{E}[f(X)] = \pm\infty$, we say that the expectation does not exist or is not well-defined.

One useful moment is the *variance*: the central second moment, where central indicates $c = \mathbb{E}[X]$. The variance indicates the amount that the random variable varies around its mean. For example, for a Gaussian distribution, if the variance σ^2 is large, then the Gaussian is very wide, indicating a non-negligible density for a broader range of points x around μ. Alternatively, if σ^2 is almost zero, then the Gaussian is concentrated tightly around μ.

We can also consider conditional expectations, and expectations for multivariate random variables. For two random variables X and Y and function $f : \mathcal{Y} \to \mathbb{R}$, the conditional expectation is

$$\mathbb{E}[f(Y)|X = x] = \begin{cases} \sum_{y \in \mathcal{Y}} f(y)p(y|x) & \text{if } Y \text{ is discrete} \\ \\ \int_{\mathcal{Y}} f(y)p(y|x)dy & \text{if } Y \text{ is continuous} \end{cases}$$

Using the identity function $f(y) = y$ results in the standard conditional expectation $\mathbb{E}[Y|x]$.

Exercise 11: Show the *law of total expectation*: $\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]]$, where the outer expectation is over X and the inner expectation is over Y. For example, if both Y and X

are discrete

$$\mathbb{E}[\mathbb{E}[Y|X]] = \sum_{x\in\mathcal{X}} p(x)\mathbb{E}[Y|X=x] = \sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} yp(y|x).$$

Hint: start first with this discrete setting, and recall the product rule $p(x,y) = p(y|x)p(x)$. □

For two random variables X and Y and $f : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$, we can also define the expectation over the joint distribution, with one variable fixed

$$\mathbb{E}\left[f(X,y)\right] = \begin{cases} \sum_{x\in\mathcal{X}} f(x,y)p(x|y) & \text{if } X \text{ is discrete} \\ \\ \int_{\mathcal{X}} f(x,y)p(x|y)dx & \text{if } X \text{ is continuous} \end{cases}$$

or over both variables

$$\mathbb{E}\left[f(X,Y)\right] = \begin{cases} \sum_{y\in\mathcal{Y}} p(y)\mathbb{E}[f(X,y)] & \text{if } Y \text{ is discrete} \\ \\ \int_{\mathcal{Y}} p(y)\mathbb{E}[f(X,y)]dy & \text{if } Y \text{ is continuous} \end{cases}$$

For example, if X is continuous and Y is discrete, this gives

$$\begin{aligned} \mathbb{E}\left[f(X,Y)\right] &= \sum_{y\in\mathcal{Y}} p(y)\mathbb{E}\left[f(X,y)\right] \\ &= \sum_{y\in\mathcal{Y}} p(y)\int_{\mathcal{X}} f(x,y)p(x|y)dx \end{aligned}$$

Exercise 12: Show that $\int_{\mathcal{X}} \left(\sum_{y\in\mathcal{Y}} f(x,y)p(x,y)\right) dx = \int_{\mathcal{X}} \mathbb{E}[f(x,Y)]p(x)dx$. □

Just as above with variance, the *covariance* is one important instance of these expected values, with $f(x,y) = (x - \mathbb{E}[X])(y - \mathbb{E}[Y])$. The expected value under this function indicates how the two variables vary together. We use specific notation for the covariance, because it is so frequently used

$$\begin{aligned} \mathrm{Cov}[X,Y] &= \mathbb{E}\left[(X - \mathbb{E}\left[X\right])\left(Y - \mathbb{E}\left[Y\right]\right)\right] \\ &= \mathbb{E}\left[XY\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right], \end{aligned}$$

with $\mathrm{Cov}[X,X] = V[X]$ being the variance of the random variable X. The *correlation* is the covariance, normalized by the standard deviation—square root of the variance—of each random variable

$$\mathrm{Corr}[X,Y] = \frac{\mathrm{Cov}[X,Y]}{\sqrt{V[X]}\sqrt{V[Y]}}.$$

The covariance can become larger, if X and Y themselves have large variance. The correlation, on the other hand, is guaranteed to be between -1 and 1, and so is an scale-invariant measure of how the variables vary together.

2.4.1 Properties of expectations and variances

Here we review some useful properties of expectations. Consider random variables, X and Y. For a constant $c \in \mathbb{R}$, it holds that:

1. $\mathbb{E}[cX] = c\mathbb{E}[X]$
2. $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ ▷ linearity of expectation
3. $\mathrm{Var}[X] \stackrel{\text{def}}{=} \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathrm{Cov}[X, X] \geq 0$. We typically write $\mathrm{Var}[X]$, instead of $\mathrm{Cov}[X, X]$.
4. $\mathrm{Var}[c] = 0$ ▷ the variance of a constant is zero
5. $\mathrm{Var}[cX] = c^2\mathrm{Var}[X]$.
6. $\mathrm{Cov}[X, Y] \stackrel{\text{def}}{=} \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}(Y)] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$
7. $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X, Y]$ ▷ when $d = m$

In addition, if X and Y are independent random variables, it holds that:

8. $\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$ for all i, j
9. $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$
10. $\mathrm{Cov}[X, Y] = 0$.

Chapter 3

An Introduction to Estimation

We first investigate a simple estimator—a sample average—with a focus on how we can have confidence in our estimate. The sample average is the average of n samples from a distribution. The sample average provides an estimate of the true mean. Depending on the distribution, we might need a larger n to get an accurate estimate. In this chapter, we provide the tools to understand the quality of our sample average, and introduce the concepts of bias, consistency, sample complexity and concentration inequalities.

3.1 Estimating the Expected Value

We assume that we get n samples from an unknown distribution p, over outcome space $\mathcal{X}$. More formally, we assume we have n random variables $X_1, \ldots, X_n$, where $\mathbb{E}[X_i] = \mu$ for some unknown mean μ. The sample average estimator is

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{3.1}$$

Intuitively, we know as n gets larger, $\bar{X}$ should get closer and closer to μ. But, how do we show this? We will use concentration inequalities, which gives us the strong law of large numbers, to show this in the next section.

First, let's ask one slightly easier question: is this estimator *unbiased*? The bias of an estimator is how far the expected value of the estimator deviates from the true value. For the sample average estimator, the bias is

$$\text{Bias}(\bar{X}) = \mathbb{E}[\bar{X}] - \mu \tag{3.2}$$

because μ is the true value we are trying to estimate. The estimator is said to be *unbiased* if the Bias is zero. The expectation $\mathbb{E}[\bar{X}]$ reflects that $\bar{X}$ is random, due to the fact that we could have observed many different plausible sets of n samples.

Example 9: To better understand why $\bar{X}$ is a random variable, let us consider a specific example where we flip a fair coin three times. The outcome of the first flip is X_1, the second flip is X_2 and the third is X_3, with $n = 3$ and $\bar{X} = \frac{1}{3}\sum_{i=1}^{3} X_i$. Because this is a fair coin, we know $p(X_i = 1) = 0.5$. A perfectly plausible sequence of flips we might see is $x_1 = 0, x_2 = 0, x_3 = 1$ with sample average $\bar{x} = 1/3$. These are all written lowercase, because they are actual outcomes we observed. Another possible sequence could be $x_1 = 1, x_2 = 0, x_3 = 1$ with sample average 2/3. The random variable X_1 represents that the first flip could be 0 or 1, with probability 0.5; the random variable X_2 represents that the second flip could be 0 or 1, with probability 0.5; and so on. Because $\bar{X}$ is the average of n random variables, it itself is also random.

We can even reason about the outcome space of $\bar{X}$ as well as the probabilities over those outcomes. The possible outcomes are $\{0, 1/3, 2/3, 1\}$. There are $2^3 = 8$ different possible sequences of coin flips: $(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0)$ and $(1,1,1)$. All of these sequences have equal probability $.5^3 = 0.125$ of occurring. This means that

$$\begin{aligned}
p(\bar{X} = 0) &= p(X_1 = 0, X_2 = 0, X_3 = 0) = 0.125 \\
p(\bar{X} = 1/3) &= p(X_1 = 0, X_2 = 0, X_3 = 1) + p(X_1 = 0, X_2 = 1, X_3 = 0) \\
&\quad + p(X_1 = 1, X_2 = 0, X_3 = 0) = 0.375 \\
p(\bar{X} = 2/3) &= p(X_1 = 0, X_2 = 1, X_3 = 1) + p(X_1 = 1, X_2 = 0, X_3 = 1) \\
&\quad + p(X_1 = 1, X_2 = 1, X_3 = 0) = 0.375 \\
p(\bar{X} = 1) &= p(X_1 = 1, X_2 = 1, X_3 = 1) = 0.125
\end{aligned}$$

We only execute the experiment once (flipping three coins), so we will see precisely one of these four outcomes. The likelihood of seeing a sample average of 1/3 or 2/3 is equally high, and it is less likely to see a sample average of 0 or 1, but both are possible.

If we had done this for 10 coin flips, instead of 3, then we would get outcome space $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ with associated probabilities

$$\begin{aligned}
p(\bar{X} = 0) &= 1/1024 \approx 10^{-3} & p(\bar{X} = 0.1) &= 45/1024 \approx 0.04 \\
p(\bar{X} = 0.2) &= 120/1024 \approx 0.12 & p(\bar{X} = 0.3) &= 210/1024 \approx 0.21 \\
p(\bar{X} = 0.5) &= 252/1024 \approx 0.25 & p(\bar{X} = 0.6) &= 210/1024 \approx 0.21 \\
p(\bar{X} = 0.7) &= 210/1024 \approx 0.21 & p(\bar{X} = 0.8) &= 120/1024 \approx 0.12 \\
p(\bar{X} = 0.9) &= 45/1024 \approx 0.04 & p(\bar{X} = 1) &= 1/1024 \approx 10^{-3}
\end{aligned}$$

□

Now, let us compute this expectation.

$$\begin{aligned}
\mathbb{E}[\bar{X}] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[X_i\right] && \triangleright\ \mathbb{E}\left[X_i\right] = \mu, \text{ by assumption} \\
&= \frac{1}{n}\sum_{i=1}^{n} \mu = \frac{1}{n} n\mu \\
&= \mu
\end{aligned}$$

So the bias is zero, because $\text{Bias}(\bar{X}) = \mathbb{E}[\bar{X}] - \mu = \mu - \mu = 0$.

We can also characterize the variance of the estimator. If the X_i are i.i.d. random

variables with variance σ^2, then

$$\begin{aligned}
\text{Var}\left[\bar{X}\right] &= \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2}\text{Var}\left[\sum_{i=1}^{n} X_i\right] \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}\left[X_i\right] && \triangleright \text{ by independence} \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sigma^2 = \frac{1}{n}\sigma^2.
\end{aligned}$$

Therefore, the variance shrinks proportionally to the number of samples. However, this does not give us enough information about how close $\bar{X}$ is to the true mean μ. For this, we use concentration inequalities in the next section.

You might wonder why we care about the bias and variance of our estimator. One reason is simply that once we start thinking about our sample average estimator as a random variable, it is useful to know statistics about the random variable, like its expectation and variance. Another reason is that it helps us reason about the expected mean-squared error of our estimator, as we discuss in Section 3.5.

3.2 Concentration Inequalities

Our goal is to obtain a confidence interval around our estimate, to obtain a measure of confidence in our estimate, and to show consistency. More specifically, we would likely be able to say that for any $\epsilon > 0$, there exists $\delta \geq 0$ such that

$$\Pr\left(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \geq \epsilon\right) \leq \delta. \tag{3.3}$$

In other words, we want a small probability δ that $\bar{X}$ deviates by ϵ from the mean $\mathbb{E}[\bar{X}]$. We want the interval given by ϵ to be small—and of course δ to be small—so that we can be confident in our estimator $\bar{X}$. This probability tells us that $\mathbb{E}[\bar{X}] \in [\bar{X} - \epsilon, \bar{X} + \epsilon]$ with high probability, that is with probability $1 - \delta$. You can see this is the case by noticing that Equation (3.3) can equivalently be written $\Pr\left(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \leq \epsilon\right) \geq 1 - \delta$ because $\Pr\left(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \leq \epsilon\right) = 1 - \Pr\left(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \geq \epsilon\right)$ and that

$$\begin{aligned}
\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \geq \epsilon &\implies -\epsilon \leq \mathbb{E}[\bar{X}] - \bar{X} \leq \epsilon \\
&\implies \bar{X} - \epsilon \leq \mathbb{E}[\bar{X}] \leq \bar{X} + \epsilon \qquad \triangleright \text{ Added } \bar{X} \text{ to all three equations.}
\end{aligned}$$

This inequality tells us that, for most sampled $\bar{x}$, $\mathbb{E}[\bar{X}] \in [\bar{x} - \epsilon, \bar{x} + \epsilon]$. Only with small probability—probability δ—will we see a $\bar{x}$ where $\mathbb{E}[\bar{X}] \notin [\bar{x} - \epsilon, \bar{x} + \epsilon]$.

If we set $\delta = 0.05$, then we say that $[\bar{X} - \epsilon, \bar{X} + \epsilon]$ is a 95% confidence interval: 95% of the time the interval we observe from a sampled $\bar{x}$ contains $\mathbb{E}[\bar{X}]$. In practice, of course, we only observe one sampled $\bar{x}$ and compute the confidence interval $[\bar{x} - \epsilon, \bar{x} + \epsilon]$. The choice of $\delta = 0.05$ gives us confidence that $\mathbb{E}[\bar{X}]$ is within this range. We often call δ the confidence level, and ϵ provides the width of the interval. Naturally, to obtain higher confidence levels, the width will be larger.

Now the question is how we find ϵ, for a chosen δ. Concentration inequalities let us characterize δ for any given ϵ, or ϵ for a given δ. We will discuss two common concentration inequalities: Hoeffding's inequality and Chebyshev's inequality. The second applies to more general settings. Let's start with Hoeffding's inequality. Assume we have independent and identically distributed (i.i.d.) bounded random variables $X_1, \ldots, X_n$, such that $a \leq X_i \leq b$ for some $a, b \in \mathbb{R}$. Then Hoeffding's inequality states that for any $\epsilon > 0$

$$\Pr(|\bar{X} - \mathbb{E}[\bar{X}]| \geq \epsilon) \leq 2\exp\left(-\frac{2n\epsilon^2}{(b-a)^2}\right).$$

For a given ϵ, we have $\delta = 2\exp(-2n\epsilon^2/(b-a)^2)$. In many cases, we would actually like to determine the interval around $\mathbb{E}[\bar{X}]$, for some confidence level δ. We can solve for ϵ in terms of δ, to get

$$\delta = 2\exp(-2n\epsilon^2/(b-a)^2) \qquad \implies \epsilon = (b-a)\sqrt{\frac{\ln(2/\delta)}{2n}}.$$

We get that with probability $1-\delta$, $\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \leq \epsilon = (b-a)\sqrt{\frac{\ln(2/\delta)}{2n}}$.

Example 10: Let's assume you have n i.i.d. random variables, with $a = 0$ and $b = 1$. Imagine you get $n = 30$ samples, with $\bar{X} = 0.6$. Now you want to get a 95% confidence interval around the true mean, i.e., $\delta = 0.05$. Then the resulting interval, using Hoeffding's inequality, has

$$\epsilon = (1-0)\sqrt{\frac{\ln(2/0.05)}{2 \times 30}} \approx 0.248$$

If instead we only require a lower confidence level, of $\delta = 0.2$, to get a 80% confidence interval, then we would have

$$\epsilon = (1-0)\sqrt{\frac{\ln(2/0.2)}{2 \times 30}} \approx 0.196.$$

This interval is a bit smaller, because we only need to say that the true expected value μ satisfies $\mu \in [\bar{X} - \epsilon, \bar{X} + \epsilon]$ with 80% probability, rather than with 95% probability. □

Hoeffding's inequality assumes bounded random variables, but there are other concentration inequalities for unbounded random variables. *Chebyshev's inequality* let's us say that, for i.i.d. random variables $X_1, \ldots, X_n$ with variance σ^2,

$$\Pr(|\bar{X} - \mathbb{E}[\bar{X}]| \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2}. \tag{3.4}$$

Again, we can solve for ϵ in terms of δ, to get

$$\delta = \frac{\sigma^2}{n\epsilon^2} \qquad \implies \epsilon = \sqrt{\frac{\sigma^2}{\delta n}}.$$

For this setting, where we know the variance, but the variables are not bounded between some a and b, we still get an interval proportional to $\sqrt{1/n}$. Notice that the conditions for Chebyshev's inequality are actually less stringent, because the variance of any random variable bounded between $[a, b]$ is at most $\sigma^2 = \frac{1}{4}(b-a)^2$. So, Chebyshev's can be applied to such random variables, by using this upper bound on the variance. However, Hoeffding's bound is a better choice, since it gives a tighter bound.

3.3 Consistency

Chebyshev's inequality let's us easily show *consistency* of the sample average estimator. Consistency means that

$$\bar{X} \to \mu \quad \text{as } n \to \infty. \tag{3.5}$$

As n gets larger, $\epsilon = \sqrt{\frac{\sigma^2}{\delta n}}$ gets smaller and smaller. In fact,

$$\sqrt{\frac{\sigma^2}{\delta n}} \to 0 \quad \text{as } n \to \infty. \tag{3.6}$$

This means that for arbitrarily small δ, $\bar{X} \to \mathbb{E}[\bar{X}]$ as $n \to \infty$. Because the sample average is unbiased, we know $\mathbb{E}[\bar{X}] = \mu$, and so $\bar{X} \to \mu$. This convergence in probability is also called the (weak) *law of large numbers*.

3.4 Rate of Convergence and Sample Complexity

The sample complexity n is the number of samples needed to obtain an ϵ accurate estimate. Our goal is to make the sample complexity small, so that we can get a good estimate with as few samples as possible—to be data efficient. The sample complexity is determined both by the properties of the data and by our estimator. We can improve the sample complexity by using smarter estimators, but will inherently have higher sample complexity for certain types of data. For example, if the data has high variance, the bound above tells us that we need more samples to obtain an accurate estimate. But, we can reduce the sample complexity if we could bias or initialize our sample average estimate to be closer to the true mean.

The convergence rate indicates how quickly the error in our estimate decays, in terms of the number of samples. For example, using Chebyshev's inequality, we obtained a convergence rate of $O(1/\sqrt{n})$:

$$\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \leq \sqrt{\frac{\sigma^2}{\delta n}} \quad \text{with high probability } 1 - \delta.$$

This concentration inequality—Chebyshev's inequality—makes few assumptions about the random variables. For example, it does not make any distributional assumptions about each X_i. We can actually reduce the sample complexity, by making stronger assumptions on the X_i.

To see why, let's revisit Equation (3.3), but now assume that the X_i are i.i.d. Gaussian random variables, with variance σ^2 and unknown mean μ. We know that $\bar{X} - \mu$ is zero-mean Gaussian with variance σ^2/n. We can look at the tails of the Gaussian to determine that, say for $\delta = 0.05$, $\epsilon = 1.96\sigma/\sqrt{n}$

$$\Pr(|\bar{X} - \mu| \geq 1.96\sigma/\sqrt{n}) = 0.95$$

Chebyshev's inequality would give a larger number of $\epsilon = 4.47\sigma/\sqrt{n}$. This is 2.28x larger than if we knew the distribution of the X_i were Gaussian, showing what we lose when we do not know the distribution and so cannot take advantage of that information to improve the confidence interval.

Notice that though sample complexity is better, the convergence rate is still $O(1/\sqrt{n})$. Does this mean we are always stuck with this rate? For many distributions, yes, but for certain distributions, we can actually get even faster convergence. For example, for independent Bernoulli X_i, the Chernoff bound—yes, yet another concentration inequality—let's us obtain a convergence rate of $O(1/n)$, which is significantly faster.

3.5 Mean-Squared Error and Bias-Variance

In the above treatment, we assumed our estimator was unbiased. But, we do not actually have to require that our estimator be unbiased. Rather, our ultimate goal is to get an estimator Y for the true mean μ, that is as close to this true value as possible. In some cases, a biased estimator might actually be closer to μ, than an unbiased one.

To see why, let's characterize the squared distance between our estimator Y and μ: $(Y-\mu)^2$. Because Y is random, this difference is random. So instead we can ask: what is the expected squared error of our estimator Y, across all possible datasets?

$$
\begin{aligned}
&\mathbb{E}[(Y-\mu)^2] \\
&= \mathbb{E}[(Y-\mathbb{E}[Y]+\mathbb{E}[Y]-\mu)^2] && \triangleright \text{ Let } b = \text{Bias}(Y) = \mathbb{E}[Y]-\mu \\
&= \mathbb{E}[((Y-\mathbb{E}[Y])+b)^2] \\
&= \mathbb{E}[(Y-\mathbb{E}[Y])^2 + 2b(Y-\mathbb{E}[Y]) + b^2] \\
&= \mathbb{E}[(Y-\mathbb{E}[Y])^2] + \mathbb{E}[2b(Y-\mathbb{E}[Y])] + \mathbb{E}[b^2] && \triangleright \text{ By linearity of expectation} \\
&= \text{Var}[Y] + 2b\mathbb{E}[(Y-\mathbb{E}[Y])] + b^2 && \triangleright \text{ By definition } \text{Var}[Y] = \mathbb{E}[(Y-\mathbb{E}[Y])^2] \\
& && \quad \text{and constants come out of expectations} \\
&= \text{Var}[Y] + 2b(\mathbb{E}[Y]-\mathbb{E}[Y]) + b^2 && \triangleright \text{ By linearity of expectation} \\
&= \text{Var}[Y] + \text{Bias}(Y)^2
\end{aligned}
$$

Therefore the mean squared error (MSE) is composed of the variance of Y and the bias of Y. The estimator is unbiased, then it is simply due to the variance of Y. If the dataset is small, for example, then the variance of Y is likely higher and the expected squared error (the MSE) is high due to insufficient samples. On the other extreme, we can get an estimator that has minimal variance: an estimator that always returns 0. This estimator, though, is clearly biased and so will have a high MSE due to a high bias. There are estimators in-between, and in some cases it can be worth having an estimator with a little bit of bias, to help reduce the variance. We see this in the next example.

Example 11: Let's define a biased estimator, $Y = \frac{1}{n+100}\sum_{i=1}^{n} X_i$. This estimator is biased because

$$
\begin{aligned}
\mathbb{E}[Y] &= \mathbb{E}\left[\frac{1}{n+100}\sum_{i=1}^{n} X_i\right] = \frac{1}{n+100}\sum_{i=1}^{n}\mathbb{E}[X_i] \\
&= \frac{n}{n+100}\mu \neq \mu \qquad \text{unless } \mu = 0 \\
\implies \text{Bias}(Y) &= \frac{n}{n+100}\mu - \mu = \frac{-100}{n+100}\mu
\end{aligned}
$$

The variance of this estimator, however, is smaller than the sample average estimator:

$$\begin{aligned}
\mathrm{Var}[Y] &= \mathrm{Var}\left[\frac{1}{n+100}\sum_{i=1}^{n} X_i\right] \\
&= \frac{1}{(n+100)^2}\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] && \triangleright\ \mathrm{Var}(cX) = c^2\mathrm{Var}(X) \\
&= \frac{1}{(n+100)^2}\sum_{i=1}^{n}\mathrm{Var}[X_i] && \triangleright\ \text{i.i.d. } X_i \\
&= \frac{n}{(n+100)^2}\sigma^2
\end{aligned}$$

You can interpret this estimator as if it saw 100 samples of zeros, and then starting seeing real data $X_1, X_2, \ldots, X_n$, i.e, $Y = \frac{1}{n+100}[\sum_{i=1}^{100} 0 + \sum_{i=1}^{n} X_i]$. It is skewed towards the value zero, which could introduce bias if $\mu \neq 0$. But, it also has lower variance since it effectively saw 100 consistent samples corresponding to zero.

Intuitively, if the true mean μ is near zero, the bias introduced is minimal and we can get significant gains from the variance reduction. If μ is far from zero, we might incur a big penalty for the bias. Assume $\sigma = 1.0$, and $n = 10$. Let's compare the MSE for the sample average estimator $\bar{X}$ and for the estimator Y.

Case 1: $\mu = 0.1$. Then, using the results for the bias and variance of the sample average estimator from Section 3.1 we have

$$\begin{aligned}
\mathrm{MSE}(\bar{X}) &= \mathbb{E}[(\bar{X} - \mu)^2] \\
&= \mathrm{Var}[\bar{X}] + \mathrm{Bias}(\bar{X})^2 && \triangleright\ \mathrm{Bias}(\bar{X}) = 0 \\
&= \mathrm{Var}[\bar{X}] && \triangleright\ \mathrm{Var}(\bar{X}) = \frac{\sigma^2}{n} \\
&= \tfrac{1}{10} && \triangleright\ \sigma = 1.0,\ n = 10 \\
\mathrm{MSE}(Y) &= \mathbb{E}[(\bar{Y} - \mu)^2] \\
&= \mathrm{Var}[Y] + \mathrm{Bias}(Y)^2 \\
&= \frac{n}{(n+100)^2}\sigma^2 + \left(\frac{100}{n+100}\mu\right)^2 \\
&= \frac{10}{(110)^2} + \left(\tfrac{100}{110}0.1\right)^2 \approx 9 \times 10^{-3}
\end{aligned}$$

In this case, where μ is not too far from 0, the MSE of Y is much lower than the MSE of the sample average estimator.

Case 2: $\mu = 5$ gives

$$\mathrm{MSE}(\bar{X}) = \mathrm{Var}[\bar{X}] = \frac{1}{10}$$

which is the same because $\bar{X}$ is unbiased so its MSE does not depend on the value of μ, and

$$\begin{aligned}
\mathrm{MSE}(Y) &= \mathrm{Var}[Y] + \mathrm{Bias}(Y)^2 \\
&= \frac{n}{(n+100)^2}\sigma^2 + \left(\frac{100}{n+100}\mu\right)^2 \\
&= \frac{10}{(110)^2} + \left(\tfrac{100}{110}5\right)^2 \approx 20.7
\end{aligned}$$

In this case, where μ is far from 0, the MSE of Y is much higher than the sample average estimator. The amount of bias introduced is higher, than the gains we obtained from having a lower variance. □

This example illustrates an important phenomenon in machine learning. We often inject some amount of prior knowledge into our learning systems. That prior knowledge often biases the solution, since it changes the solution but is not based on the actual given data. If that prior knowledge is helpful, it can often help constrain the space—and reduce variance—and so make the MSE smaller, even if some bias is incurred. On the other hand, if that prior knowledge is wrong, then it can incur too much bias. In the example above, the prior knowledge was that the mean was likely close to zero. If that prior assumption is correct, then we can get a good estimator with much less data. If the prior assumption is wrong, though, it can significantly degrade performance.

The above example was all about the small sample setting. Once n gets very big, the bias in Y is washed away.

Exercise 13: Show that Y is consistent. □

Chapter 4

Introduction to Optimization

Much of machine learning deals with learning functions by finding the optimal function according to an objective. For example, one may be interested in finding a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes the squared differences to some targets for all the samples: $\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2$. To find such a function, you need to have a basic grasp of optimization techniques.

In this chapter, we discuss basic optimization tools, particularly focused on smooth objectives. Many of the algorithms in machine learning rely on a simple approach: gradient descent. We first discuss how to minimize objectives using both first and second-order gradient descent. This overview covers only a small part of optimization, but fortunately, many machine learning algorithms are based on these simple optimization approaches.

4.1 Discrete and Continuous Optimization Problems

A basic optimization goal is to select a set of parameters $\mathbf{w} \in \mathcal{W}$ to minimize a given objective function $c : \mathcal{W} \to \mathbb{R}$

$$\min_{\mathbf{w} \in \mathcal{W}} c(\mathbf{w})$$

where $\mathcal{W}$ is the set of all possible parameters. For example, to obtain the parameters $\mathbf{w} \in \mathbb{R}^d$ for linear regression that minimizes the squared differences, we use $c(\mathbf{w}) = \sum_{i=1}^{n}(\langle \mathbf{x}_i, \mathbf{w} \rangle - y_i)^2$, for dot product

$$\langle \mathbf{x}_i, \mathbf{w} \rangle = \sum_{j=1}^{d} x_{ij} w_j.$$

The parameter set is $\mathcal{W} = \mathbb{R}^d$. This example is a *continuous optimization* problem, because $\mathbf{w}$ is a real-valued variable for a continuous objective. (See Section 1.2 for a reminder about the definition of a continuous function).

We use the term objective here, rather than error, since error has an explicit connotation that the function is inaccurate. Later we will see that objectives will include both error terms—indicating how accurately they recreate data—as well as terms that provide other preferences on the function. Combining these terms with the error produces the final objective we would like to minimize. For example, for linear regression, we will optimize a regularized objective, $c(\mathbf{w}) = \sum_{i=1}^{n}(\langle \mathbf{x}_i, \mathbf{w} \rangle - y_i)^2 + w_2^2$ where the second term encodes a preference for smaller coefficients w_2.

A *discrete optimization* problem is one where the set of elements $\mathcal{W}$ is a finite—also called discrete—set. For example, we may want to find the best of three possible parameters $\mathcal{W} = \{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$ that minimizes the objective c. This optimization is straightforward: we simply test each of the three values and return the one that gives the lowest c, namely minimal $c(\mathbf{a})$, $c(\mathbf{b})$ or $c(\mathbf{d})$. If $\mathcal{W}$ is a very large finite set, however, then this can become

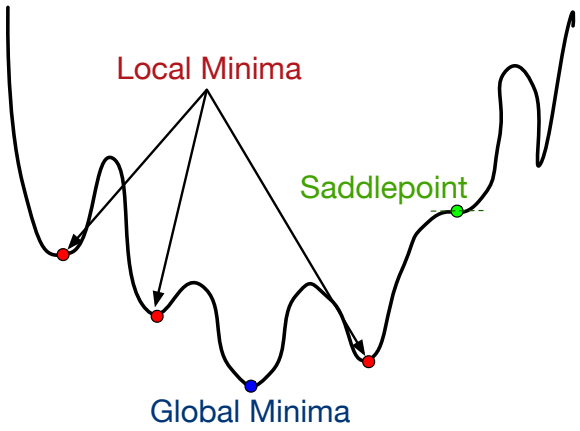


Figure 4.1: Stationary points on a smooth function surface: local minima, global minima and saddlepoints.

expensive; we may even prefer to reformulate the problem as a continuous optimization, if possible. For these notes, we assume that we either have an easy discrete optimization problem, where we can enumerate the items, or that we are dealing with a continuous optimization problem.

Remark about maximizing versus minimizing: We have so far discussed the goal of minimizing an objective. An equivalent alternative is to maximize the negative of this objective.

$$\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) = \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} -c(\mathbf{w})$$

where argmin returns $\mathbf{w}$ that produces the minimum value of $c(\mathbf{w})$ and argmax returns $\mathbf{w}$ that produces the maximum value of $-c(\mathbf{w})$. The actual min and max values are not the same, since for a given optimal solution, $c(\mathbf{w}) \neq -c(\mathbf{w})$. We opt to formulate each of our optimizations as a minimization, simply by convention. It would be equally valid, however, to formulate the optimizations as maximizations.

4.2 Stationary Points for Continuous Optimization Problems

Now back to the goal to find $\mathbf{w}$ that minimizes a continuous objective. The most straightforward, naive solution could be to do a random search: generate random $\mathbf{w}$ and check $c(\mathbf{w})$. If any newly generated $\mathbf{w}_t$ on iteration t outperforms the previous best solution $\mathbf{w}$, in that $c(\mathbf{w}_t) < c(\mathbf{w})$, then we can set $\mathbf{w}_t$ to be the new optimal solution. Because we assume our objectives are continuous, however, we can take advantage of this smoothness to design better search strategies. In particular, for smooth functions, we will be able to use gradient descent, which we describe in the next section.

Our goal is to exploit the smoothness of our objectives to efficiently search for stationary points: points $\mathbf{w}$ where the derivative—also called the gradient—is zero. Consider first the univariate case. The derivative tells us the rate of change of the function surface at a point w. When the derivative of the objective is zero at $w \in \mathbb{R}$, i.e., $\frac{d}{dw}c(w) = 0$, this means that locally the function surface is flat. Such points correspond to local minima, local maxima

and saddlepoints, as shown in Figure 4.1. A local minima is a global minima if it obtains the minimum on the objective.

For example, assume again that we are doing linear regression, with only one feature and so only one weight $w \in \mathbb{R}$. The derivative of the objective $c(w) = \sum_{i=1}^{n}(x_i w - y_i)^2$ is

$$\begin{aligned}\frac{d}{dw}c(w) &= \frac{d}{dw}\sum_{i=1}^{n}(x_i w - y_i)^2 \\ &= \sum_{i=1}^{n}\frac{d}{dw}(x_i w - y_i)^2 \\ &= \sum_{i=1}^{n} 2(x_i w - y_i)x_i\end{aligned}$$

where the last step follows from the chain rule. Our goal is to find w such that $\frac{d}{dw}c(w) = 0$; once we find such a stationary point, we can then determine if it is a local minimum, local maximum or saddlepoint.

Sometimes we can infer what type of stationary point we have simply from properties of the objective. In particular, if the objective is *convex*, then we know that the stationary point is a global minima. A function $c : \mathbb{R}^d \to \mathbb{R}$ is said to be convex if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$ and $t \in [0, 1]$,

$$c(t\mathbf{w}_1 + (1-t)\mathbf{w}_2) \leq tc(\mathbf{w}_1) + (1-t)c(\mathbf{w}_2) \tag{4.1}$$

This definition means that when we draw a line between any two points on the function surface, the function values between these two points all lie below this line. Intuitively, this means the function surface is shaped like a cup, and so the stationary point (or points) are all at the bottom of the cup and are global minima. A corresponding definition is a concave function, which is precisely the opposite: all points lie above the line. For any convex function c, the negative of that function $-c$ is a concave function.

The second derivative test tells us locally if the stationary point is a local minimum, local maximum or if it is inconclusive. Namely, the test is

1. If $c''(w_0) > 0$ then w_0 is a local minimum.
2. If $c''(w_0) < 0$ then w_0 is a local maximum.
3. If $c''(w_0) = 0$ then the test is inconclusive: we cannot say which type of stationary point we have and it could be any of the three.

To understand this test, notice that the second derivative tells us the local curvature of the function. It tells us how the derivative is changing. If the slope of the derivative $c'(w_0)$ is positive at w_0, namely $c''(w_0) > 0$, then we know that the derivative is increasing; if it is negative, then it is decreasing.

Example 12: Let us consider an example to understand this better, in Figure 4.2. Consider a sin curve $\sin(w)$ and the point halfway between the bottom and top of the hill. At one these in-between points, say $w = 0$, the derivative is maximally positive: it is $\cos(0) = +1$. As we increase w, the derivative starts to decrease until it is zero at the top of the hill, at $w = \pi/2$. Then it flips and gets more and more negative until it reaches $w = \pi$ with derivative maximally negative at $\cos(\pi) = -1$. In this region between $[0, \pi]$, the derivative is

constantly decreasing and the second derivative is negative. Then, the derivative begins to increase from its maximally negative point $\cos(\pi) = -1$, and becomes less and less negative until reaching the bottom of the hill for $w = 3\pi/2$ and becoming zero. Then again the slope flips and starts to get more and more positive until reaching 2π. In this region between $[\pi, 2\pi]$ the derivative is constantly increasing and the second derivative is positive.

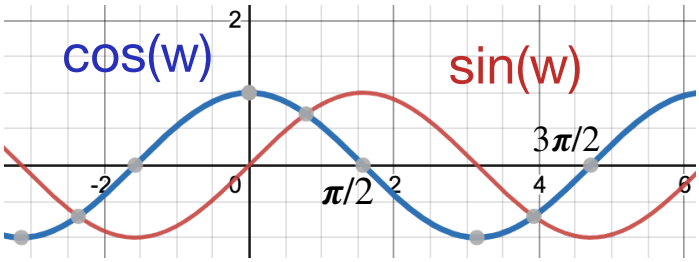


Figure 4.2: Visualizing the behavior of the derivative for the red function $c(w) = \sin(w)$, where the derivative iis in blue $c'(w) = \cos(w)$.

In the first region $[0, \pi]$ the function is concave and in the second region $[\pi, 3\pi/2]$ it is convex. Locally, the stationary point in a concave region will be a maxima; for a convex region, it will be a minima. The second derivative tells us this local curvature. □

4.3 Reaching Stationary Points with Gradient Descent

The key idea behind gradient descent is to approximate the function with a Taylor series approximation. This approximation facilitates computation of a descent direction locally on the function surface. We begin by considering the univariate setting, with $w \in \mathbb{R}$. A function $c(w)$ in the neighborhood of point w_0, can be approximated using the Taylor series

$$c(w) = \sum_{n=0}^{\infty} \frac{c^{(n)}(w_0)}{n!}(w - w_0)^n,$$

where $c^{(n)}(w_0)$ is the n-th derivative of function $c(w)$ evaluated at point w_0. This assumes that $c(w)$ is infinitely differentiable, but in practice we will take such polynomial approximations for a finite n. A *second-order* approximation to this function uses the first three terms of the series as

$$c(w) \approx \hat{c}(w) = c(w_0) + (w - w_0)c'(w_0) + \frac{1}{2}(w - w_0)^2 c''(w_0).$$

A stationary point of this $\hat{c}(w)$ can be easily found by finding the first derivative and setting it to zero

$$c'(w) \approx c'(w_0) + (w - w_0)c''(w_0) = 0.$$

Solving this equation for w gives us

$$w_1 = w_0 - \frac{c'(w_0)}{c''(w_0)}.$$

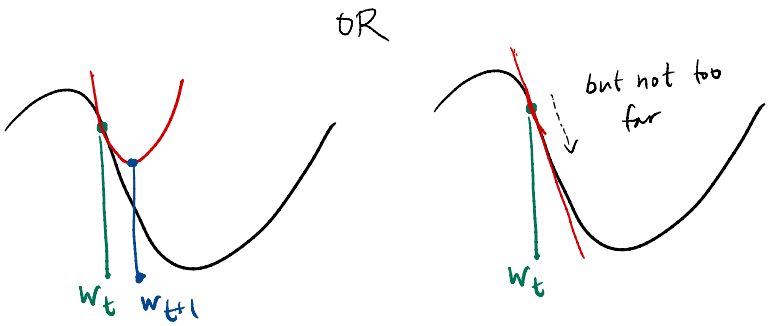


Figure 4.3: Gradient descent, using a local quadratic approximation at a point with the Taylor series. The first figure, on the left, show how the new point is found by going to the minimum of the local quadratic function. The figure on the right provides the intuition of taking a small step in the negative direction of the gradient, for first-order gradient descent.

Locally, this new w_1 will be an improvement on w_0, and will be a stationary point of this local approximation $\hat{c}$. Moving (far enough) from w_0, however, makes this local second-order Taylor series inaccurate. We would need to check the local approximation at this new point w_1, to determine if we can further improve locally. Therefore, to find the optimal w, we can iteratively apply this procedure

$$w_{t+1} = w_t - \frac{c'(w_t)}{c''(w_t)}. \tag{4.2}$$

constantly improving w_i until we reach a point where the derivative is zero, or nearly zero. This method is called the Newton-Raphson method, or second-order gradient descent.[1] It is depicted in Figure 4.3.

In first-order gradient descent, the approximation is worse, where we no longer use the true second derivative. Instead, we guess or approximate the second derivative by picking a value η_t such that $\frac{1}{\eta_t} \approx c''(w_t)$. This new term η_t is called the *stepsize*, because it dictates how far we step in the direction of the gradient. Namely, if we solve for $w_{t+1} = \text{argmin}_w\, c(w_t) + (w - w_t)c'(w_t) + \frac{1}{2\eta_t}(w - w_t)^2$, we get the update

$$w_{t+1} = w_t - \eta_t c'(w_t). \tag{4.3}$$

From this, one can see that, given access to the second derivative, a reasonable choice for the stepsize is $\eta_t = \frac{1}{c''(w_t)}$.

We can similarly obtain such rules for multivariate variables. Gradient descent for $c : \mathbb{R}^d \to \mathbb{R}$ consists of the update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t). \tag{4.4}$$

where

$$\nabla c(\mathbf{w}_t) = \left(\frac{\partial c}{\partial w_1}(\mathbf{w}_t), \frac{\partial c}{\partial w_2}(\mathbf{w}_t), ..., \frac{\partial c}{\partial w_d}(\mathbf{w}_t) \right) \qquad \in \mathbb{R}^d$$

[1] You may notice a theme where I prefer descriptive algorithm names, rather than those based on the names of people. We will call this algorithm second-order gradient descent.

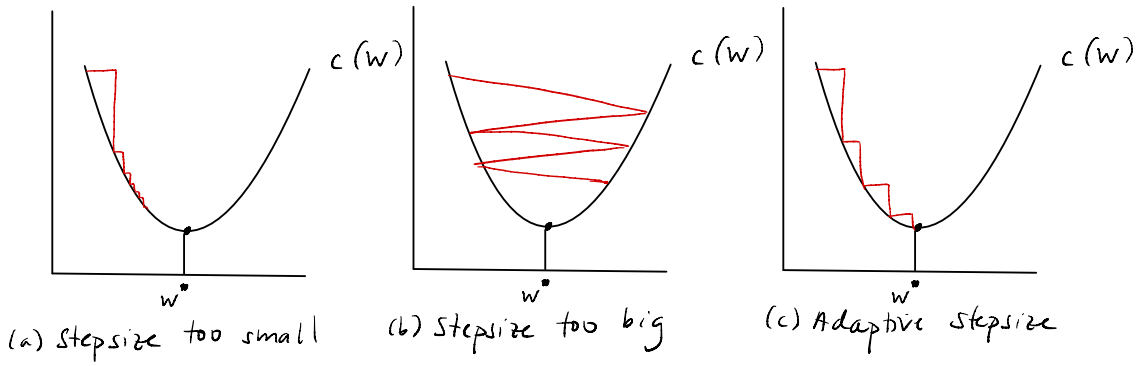


Figure 4.4: Different optimization paths, due to different stepsize choices. In (a), for fixed small stepsize, the first step is reasonably large, because the gradient is large. As the gradient gets smaller, as we get closer to w^*, the stepsize is too small. In (b), for a fixed big stepsize, this would avoid the issue in (a) where the stepsize is too small once we get closer to w^*. But, for the first gradient, such a big stepsize is too big, causing the update to jump all the way to a w on the other size of w^*. In (c), we have an adaptive stepsize—such as might be given by using $\eta_t = 1/c''(w_t)$—that is smaller on the first step and gets larger as the gradient gets smaller.

is the gradient of function c evaluated at $\mathbf{w}_t$. Each *partial derivative* $\frac{\partial c}{\partial w_j}(\mathbf{w}_t)$ indicates how the function c changes if all the variables in $\mathbf{w}_t$ are held constant except for the j element. As in the single variable setting, the gradient gives a descent direction where stepping (a sufficiently small step) in that direction will decrease the function value. If the gradient is zero, then we are already at a stationary point. We can also similarly derive second-order updates for the multivariate setting, but will only use first-order approaches here with well chosen stepsizes.

4.4 Selecting the Step-size

An important part of (first-order) gradient descent is to select the step-size. If the step-size is too small, then many iterations are required to reach a stationary point (Figure 4.4(a)). If the step-size is too large, then you are likely to oscillate around the minimum (Figure 4.4 (b)). What we really want is an adaptive step-size (Figure 4.4 (c)), that adjusts to the magnitude of the gradient.

The basic method to obtain adaptive step-sizes is to use *line search*. The idea springs from the following goal: we would like to obtain the optimal step-size according to

$$\min_{\eta \in \mathbb{R}^+} c(\mathbf{w}_t - \eta \nabla c(\mathbf{w}_t))$$

The solution to this optimization corresponds to the best scalar stepsize we could select, for the current point $\mathbf{w}_t$ with descent direction $-\nabla c(\mathbf{w}_t)$. Solving this optimization would be too expensive; however, we can find approximate solutions quickly. One natural choice is to use a backtracking line search, that tries the largest reasonable stepsize $\eta_{\max}$, and then reduces it until the objective is decreased. The idea is to search along the line of possible $\eta \in (0, \eta_{\max}]$, with the intuition that a large step is good—as long as it does not overshoot.

If it does overshoot, then the stepsize was too large, and should be reduced. The reduction is typically according to the rule $\tau\eta$ for some $\tau \in [0.5, 0.9]$. For $\tau = 0.5$, the stepsize reduces more quickly—halves on each step of the backtracking line search; for $\tau = 0.9$, the search more slowly backtracks from $\eta_{\max}$. As soon as a stepsize is found that decreases the objective, it is accepted. We then obtain a new $\mathbf{w}_t$, again compute the gradient and start the line search once again from $\eta_{\max}$.

This basic line search provides some intuition for our goal in adapting the stepsize. One can, of course, imagine other strategies for selecting the stepsize. We might prefer more efficient choices, even if they are heuristic. For example, for $\mathbf{g}_t \stackrel{\text{def}}{=} \nabla c(\mathbf{w}_t)$ with $g_{t,j}$ the j-th index into $\mathbf{g}_t$, a simple strategy to be robust to big gradients is to use the stepsize

$$\eta_t = (1 + \sum_{j=1}^{d} |g_{t,j}|)^{-1}.$$

This heuristic uses the magnitude of the gradient on the denominator, plus 1 to ensure that our stepsize is never too large. For example, if $\mathbf{g}_t$ is very close to zero, then $(\sum_{j=1}^{d} |g_{t,j}|)^{-1}$ can be very large. To avoid this, we add a small constant in the denominator. Such a constant can be a tuned ϵ value $(\epsilon + \sum_{j=1}^{d} |g_{t,j}|)^{-1}$, but for simplicity we simply set $\epsilon = 1$.

Algorithm 1: Line Search($\mathbf{w}_t, c, \mathbf{g} = \nabla c(\mathbf{w}_t)$)

Optimization parameters: $\eta_{\max} = 1.0, \tau = 0.7, \text{tolerance} \leftarrow 10^{-4}$
$\eta \leftarrow \eta_{\max}$
$\mathbf{w} \leftarrow \mathbf{w}_t$
obj $\leftarrow c(\mathbf{w})$
while number of backtracking iterations is less than maximum iterations **do**
    $\mathbf{w} \leftarrow \mathbf{w}_t - \eta\mathbf{g}$
    // Ensure improvement is at least as much as tolerance
    If $c(\mathbf{w}) <$ obj - tolerance **then break**
    // Else, the objective is worse and so we decrease stepsize
    $\eta \leftarrow \tau\eta$
if maximum number of iterations reached **then**
    // Could not improve solution
    return $\mathbf{w}_t, \eta = 0$
return $\mathbf{w}, \eta$

4.5 Testing for Optimality and Solution Uniqueness

Recall that our ultimate goal is to find a solution to our optimization problem. Finding a stationary point, therefore, is only a first step. After obtaining a stationary point, we then have to check: (a) is it a local minimum, maximum or saddlepoint and (b) if it is a local minimum, can we further conclude it is a global minimum.

As mentioned above, for a convex function, the stationary point(s) will all be global minima. Therefore, regardless of where we start our gradient descent, with appropriately chosen stepsize and sufficient iterations, we will reach an optimal solution.

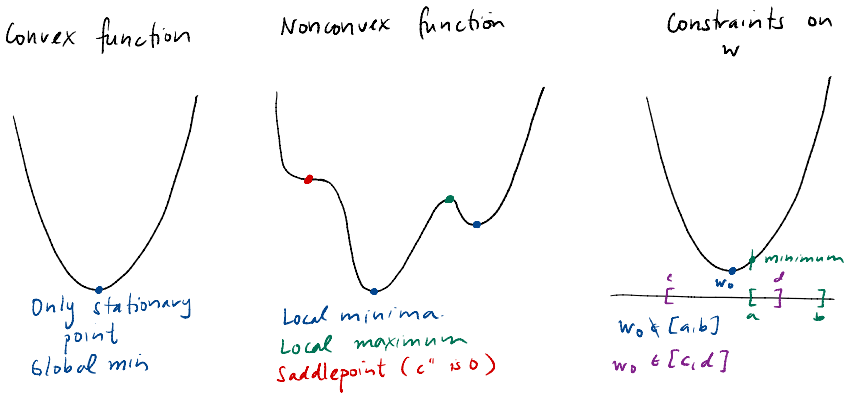


Figure 4.5: The second derivative test and reasoning about optimality. For (a), for a convex function, all stationary points are global minima. If this function had a flat region at the minimum, the second-derivative test would be inconclusive, because the second derivative would be zero (flat curvature). But, we would still know that the stationary point was one of many equivalent global minima, because the function is convex. For (b), we can use the second derivative test to distinguish between local minima and maxima for nonconvex functions. But, we cannot say for sure if our local minimum is a global minimum; the second derivative test returns the same conclusion for both of these minima. In this case, a second derivative of zero indicates a saddlepoint, but the test is generally inconclusive because flat regions that are minimal or maximal regions also have flat curvature.

If we are unsure about the convexity of the function, then we turn to the second derivative test. Recall that the second derivative test tells us locally if the stationary point is a local minimum, local maximum or if it is inconclusive. We can conclude that, if we have an unconstrained optimization, a stationary point is a global minimum if (a) the second derivative test tells us it is a local minimum and (b) we have one stationary point.

In some cases we will have constraints on the variables we are optimizing. A common one—and the only one we address in these notes—is lower and upper bounds on the variables: $w \in [a, b]$. If the stationary point w_0 satisfies $w_0 \in [a, b]$ and the function is convex, then the boundary points do not correspond to a solution. If the stationary point is outside the feasible set, then one of the boundary points might be a solution. This is visualized in Figure 4.5.

Uniqueness of the solution. We often care if there is more than one solution to our optimization problem. In some cases, we care about *identifiability*, which means we can identify the true solution. If there is more than one solution, one might consider that the problem is not precisely posed. For some problems, it is important or even necessary to have identifiability (e.g., estimating the percentage of people with a disease) whereas for others we simply care about finding a suitable (predictive) function f that reasonably accurately predicts the targets, even if it is not the unique such function. We will not consider identifiability further in this document, but it is important to be cognizant of if your objective has multiple solutions.

Equivalence under a constant shift Adding or multiplying by a constant $a \neq 0$ does not change the solution

$$\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} a\ c(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + a.$$

You can see why by taking the gradient of all three objectives and noticing that the gradient is zero under the same conditions

$$\nabla a\ c(\mathbf{w}) = 0 \iff a\nabla c(\mathbf{w}) = 0 \iff \nabla c(\mathbf{w}) = 0$$

and

$$\nabla(c(\mathbf{w}) + a) = 0 \iff \nabla c(\mathbf{w}) = 0.$$

Chapter 5

Formalizing Parameter Estimation

In probabilistic modeling, we are typically presented with a set of observations and the objective is to find a model that well approximates the true underlying model that generated the data. For example, we may get a dataset of commute times. These commute times might actually be sampled i.i.d. from a Gamma distribution, with parameters α^* and β^*. Namely, in the limit, as we sample more and more commute times, the density over commute times will perfectly match a Gamma(α, β). Intuitively, we should be able to identify these parameters—or *estimate* them—from a sufficiently large dataset. In this section, we talk about how to formalize this parameter estimation problem.

The key steps involve 1) picking the distribution and parameters you will estimate (e.g., Gamma with parameters α and β), 2) writing down the optimization that formalizes which parameters are the "best" choice (e.g., most likely) given the observed data. As you will see, once we do Step 1, there are relatively standard choices for Step 2: maximum a posteriori estimation (MAP), maximum likelihood estimation (MLE) and Bayesian estimators. Step 1 is not as straightforward. It is a mixture of expert knowledge and experience. In this chapter, we will focus on Step 2, and conclude the chapter with a discussion about Step 1.

5.1 Maximum Likelihood Estimation

Imagine you observe a dataset of observations $\mathcal{D} = \{x_i\}_{i=1}^n$. The data is drawn from some true distribution p^*, but that distribution is unknown to you. Instead, all you know is that the distribution is in a set of possible distributions, $\mathcal{F}$, sometimes called the *hypothesis space* or function class. For example, $\mathcal{F}$ could be the family of all univariate Gaussian distributions:

$$\mathcal{F} = \{\mathcal{N}(\mu, \sigma^2) \mid \text{ for any } \mu \in \mathbb{R} \text{ and } \sigma \in \mathbb{R}^+\}.$$

The true distribution has parameters μ^* and σ^*. Using the data, we would like to find μ and σ as close to these as possible.

One reasonable objective is to pick the function (parameters) that make the data the most likely. This is called *maximum likelihood*, and is written

$$f_{\text{MLE}} = \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)$$

where $p(\mathcal{D}|f)$ is called the *likelihood* of the data given the model. If the data has low likelihood for a distribution given by f, then it is unlikely that that f corresponds to the true parameters that generated the data. Conversely, f for which the data is the most likely is more likely to correspond to f^*, especially if we have a lot of data. Maximum likelihood estimation (MLE) is also motivated by the connection to MAP, described in the next section.

Note that we use words *model*, which is a function, and its *parameters*, which are the coefficients of that function, somewhat interchangeably. For example, above, we could have equivalently considered $\mathcal{F} = \{(\mu \in \mathbb{R}, \sigma \in \mathbb{R}^+)\}$. We will typically reason directly about the parameter space rather than indirectly about the models or probabilities that they parameterize.

Example 13: Suppose data set $\mathcal{D} = \{2, 5, 9, 5, 4, 8\}$ is an i.i.d. sample from a Poisson distribution with a fixed but unknown parameter λ_0. The probability mass function of a Poisson distribution is expressed as $p(x|\lambda) = \lambda^x e^{-\lambda}/x!$, with some parameter $\lambda \in \mathbb{R}^+$. The MLE optimization is

$$\lambda_{\text{MLE}} = \operatorname*{argmax}_{\lambda \in (0,\infty)} p(\mathcal{D}|\lambda). \tag{5.1}$$

We can write the likelihood function as

$$\begin{aligned} p(\mathcal{D}|\lambda) = p(\{x_i\}_{i=1}^n|\lambda) &= \prod_{i=1}^n p(x_i|\lambda) \\ &= \prod_{i=1}^n \lambda^{x_i} e^{-\lambda}/x_i! && \triangleright \prod_{i=1}^n \lambda^{x_i} = \lambda^{\sum_{i=1}^n x_i} \text{ because } \lambda^a\lambda^b = \lambda^{a+b} \\ &= \frac{\lambda^{\sum_{i=1}^n x_i} \cdot e^{-n\lambda}}{\prod_{i=1}^n x_i!}. \end{aligned}$$

where the probability breaks up into individual probabilities of each x_i because the data is i.i.d. (the random variables are independent). □

This example is quite abstract, so we also consider a much simpler example with coins.

Example 14: Assume we are given a coin with an unknown probability of seeing a 1 (a heads). Our goal is to estimate this parameter w, which we sometimes call the bias of the coin. The manufacturer that gave us the coin assures us that the bias is one of $\mathcal{F} = \{0.3, 0.5, 0.8\}$. Imagine we obtain a dataset of n coin flips, $\mathcal{D} = \{0, 1, 0, 1, 1, \ldots, 1\}$ namely where $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 1, \ldots, x_n = 1$. Given w, the distribution over an outcome is $p(x_i|w) = w^{x_i}(1-w)^{1-x_i}$. Similarly, we can compute

$$\begin{aligned} p(\mathcal{D}|w) = \prod_{i=1}^n p(x_i|w) &= \prod_{i=1}^n w^{x_i}(1-w)^{1-x_i} \\ &= w^{n_1}(1-w)^{n_0} \end{aligned}$$

where n_1 is the number of observed 1s and n_0 the number of observed 0s, with $n = n_1 + n_0$. Further imagine that $n = 10$ and $n_1 = 8$ and $n_0 = 2$. Then we can evaluate $p(\mathcal{D}|w)$ for all three values of w:

$$\begin{aligned} p(\mathcal{D}|w = 0.3) &= 0.3^{n_1}0.7^{n_0} \approx 3.21 \times 10^{-5} \\ p(\mathcal{D}|w = 0.5) &= 0.5^{n_1}0.5^{n_0} \approx 9.77 \times 10^{-4} \\ p(\mathcal{D}|w = 0.8) &= 0.8^{n_1}0.2^{n_0} \approx 6.71 \times 10^{-3} \end{aligned}$$

To find $\operatorname{argmax}_{w \in \{0.3, 0.5, 0.8\}} p(\mathcal{D}|w)$, we simply pick the w that gives the maximum in these three equations, which is $w = 0.8$.

If on the other hand we did not know that $w \in \{0.3, 0.5, 0.8\}$, and instead assumed $w \in [0, 1]$, we would solve for $\text{argmax}_{w \in [0,1]} p(\mathcal{D}|w)$. This is now a continuous optimization problem, and we will have to use the strategies from Chapter 4 to find the w that maximizes the likelihood. □

We can go ahead and use our optimization tools, namely gradient descent, to optimize this objective. However, the resulting gradient is annoying to compute, due to all the products. Fortunately, we have a simple work-around to specify an equivalent by easier-to-use objective: the log likelihood. The idea is simple: we transform the likelihood with the log function, which turns products into sums but does not change the relative ordering of parameters. Namely

$$\begin{aligned} f_{\text{MLE}} = \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f) &= \operatorname*{argmax}_{f \in \mathcal{F}} \ln p(\mathcal{D}|f) \\ &= \ln \prod_{i=1}^{n} p(x_i|f) = \sum_{i=1}^{n} \ln p(x_i|f). \end{aligned}$$

The last step follows from the property of logs: $\log(abc) = \log(a) + \log(b) + \log(c)$. Further, since the logarithm is a monotonically increasing function, we know that for any $a, b > 0$, that $a > b$ if and only if $\ln a > \ln b$. Therefore, relative ordering between f is preserved, since the ordering on likelihoods and log-likelihoods are the same.

Additionally, we will typically choose to formalize problems as minimization problems rather than maximization, just by convention. As discussed in Chapter 4, they are perfectly equivalent, as one just uses the negative of the function. We therefore will typically minimize the negative log-likelihood, rather than maximizing the likelihood

$$\begin{aligned} f_{\text{MLE}} &= \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f) \\ &= \operatorname*{argmax}_{f \in \mathcal{F}} \ln p(\mathcal{D}|f) \\ &= \operatorname*{argmin}_{f \in \mathcal{F}} -\ln p(\mathcal{D}|f). \end{aligned} \tag{5.2}$$

Notice that the above formulas say that the `argmax` is equal. Recall that the argmax is the argument that gives the maximum value. The value at the maximum or minimum itself, though, is different. Our goal is to identify the parameters, not this maximal or minimal value, so to us all these three objectives are equivalent.

Example 15: Now let us return to the above example, with data set $\mathcal{D} = \{2, 5, 9, 5, 4, 8\}$ sampled from a Poisson distribution with a fixed but unknown parameter λ_0. Our goal now is to find the maximum likelihood estimate of λ_0.

The probability mass function of a Poisson distribution is expressed as $p(x|\lambda) = \lambda^x e^{-\lambda}/x!$, with some parameter $\lambda \in \mathbb{R}^+$. We will estimate this parameter as

$$\lambda_{\text{MLE}} = \operatorname*{argmax}_{\lambda \in (0,\infty)} p(\mathcal{D}|\lambda). \tag{5.3}$$

First, notice that[1]

$$\begin{aligned} \ln p(x_i|\lambda) &= \ln \lambda^{x_i} e^{-\lambda}/(x_i)! \\ &= \ln \lambda^{x_i} + \ln e^{-\lambda} - \ln x_i! \\ &= x_i \ln \lambda - \lambda - \ln x_i! \\ \implies -\ln p(x_i|\lambda) &= -x_i \ln \lambda + \lambda + \ln x_i! \end{aligned}$$

The negative log-likelihood is the objective, composed of $c_i(\lambda) \stackrel{\text{def}}{=} -\ln p(x_i|\lambda)$:

$$c(\lambda) \stackrel{\text{def}}{=} -\ln p(\mathcal{D}|\lambda) = \sum_{i=1}^{n} c_i(\lambda).$$

We can compute the gradient of this objective, by computing gradient $c'(\lambda) = \sum_{i=1}^{n} c_i'(\lambda)$, as discussed in Chapter 4.

$$\begin{aligned} c'(\lambda) &= \frac{d}{d\lambda}(-x_i \ln \lambda + \lambda + \ln x_i!) \\ &= \frac{d}{d\lambda}(-x_i \ln \lambda) + \frac{d}{d\lambda}(\lambda) + \frac{d}{d\lambda}(\ln x_i!) \\ &= -x_i \frac{d}{d\lambda} \ln \lambda + 1 + 0 && \triangleright \ \frac{d}{d\lambda} \ln \lambda = \frac{1}{\lambda} \\ &= \frac{-x_i}{\lambda} + 1 \end{aligned}$$

Solving for $c'(\lambda) = 0$ gives us a stationary point for this problem

$$\begin{aligned} c'(\lambda) &= \sum_{i=1}^{n} \left(\frac{-x_i}{\lambda} + 1 \right) = -\frac{1}{\lambda} \sum_{i=1}^{n} x_i + n = 0 \\ \implies n &= \frac{1}{\lambda} \sum_{i=1}^{n} x_i \\ \implies \lambda &= \frac{1}{n} \sum_{i=1}^{n} x_i \end{aligned}$$

which is simply a sample mean. We can substitute $n = 6$ and values from $\mathcal{D}$ to compute the solution as

$$\lambda_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i = 5.5$$

This objective, for this dataset, is visualized in Figure 5.1.

Finally, if we want to ensure that this is a local minimum, rather than maximum, we can use the second derivative test. The second derivative is $c''(\lambda) = \lambda^{-2} \sum_{i=1}^{n} x_i$ which is > 0 for this λ_{MLE}. Therefore, the objective is locally convex, and so we are at a local minimum. We further know it is a global minimum, since it is the only stationary point. Note that to properly maximize this loss, we also need to ensure the constraint $\lambda \in (0, \infty)$ is enforced. Because the solution above is in the constraint set, we know we have the correct solution to Equation (5.3). □

[1] Recall that for scalars $a, b > 0$, (i) $\ln(ab) = \ln a + \ln b$ (ii) $\ln(a/b) = \ln a - \ln b$ and (iii) $\ln a^b = b \ln a$

5.2 MAP Estimation

The idea behind *maximum a posteriori* (MAP) estimation is to find the most probable model for the observed data. Given the data set $\mathcal{D}$, we formalize the MAP solution as

$$f_{\text{MAP}} = \operatorname*{argmax}_{f \in \mathcal{F}} p(f|\mathcal{D})$$

where $p(f|\mathcal{D})$ is called the *posterior distribution* of the model given the data. In discrete model spaces, $p(f|\mathcal{D})$ is the probability mass function and the MAP estimate is exactly the most probable model. Its counterpart in continuous spaces is the model with the largest value of the posterior density function. This objective more explicitly considers our uncertainty about the model f, because we reason about the distribution over f.

To calculate the posterior distribution we start by applying Bayes rule as

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}, \tag{5.4}$$

where $p(\mathcal{D}|f)$ is called the *likelihood* function, $p(f)$ is the *prior* distribution of the model, and $p(\mathcal{D})$ is the *marginal* distribution of the data. Notice that we use $\mathcal{D}$ for the observed data set, but that we usually think of it as a realization of a multidimensional random variable D drawn according to some distribution $p(\mathcal{D})$. Using the formula of total probability, we can express $p(\mathcal{D})$ as

$$p(\mathcal{D}) = \begin{cases} \sum_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f) & f : \text{discrete} \\ \\ \int_{\mathcal{F}} p(\mathcal{D}|f)p(f)df & f : \text{continuous} \end{cases}$$

Therefore, the posterior distribution over f can be fully described using the likelihood and the prior. Computing this prior, though, can be prohibitively expensive. For example, it could require the estimation of an integral over all possible models f.

Fortunately, finding f_{MAP} can be greatly simplified because $p(\mathcal{D})$ in the denominator does not affect the solution. This is because $p(\mathcal{D})$ is the same regardless of f in the maximization, and so scaling by $p(\mathcal{D})$ does not change the relative ordering

$$\begin{aligned} \max_{f \in \mathcal{F}} \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})} &= \frac{1}{p(\mathcal{D})} \max_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f) \\ \implies \operatorname*{argmax}_{f \in \mathcal{F}} \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})} &= \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f). \end{aligned}$$

For this reason, we often write

$$\begin{aligned} p(f|\mathcal{D}) &= \frac{p(\mathcal{D}|f) \cdot p(f)}{p(\mathcal{D})} \\ &\propto p(\mathcal{D}|f) \cdot p(f), \end{aligned}$$

where $\propto$ is the proportionality symbol. And, we find the MAP solution by solving the following optimization problem

$$f_{\text{MAP}} \stackrel{\text{def}}{=} \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f).$$

Notice that again we can apply the log without changing the relative order and rewrite this as a minimization by taking the negative

$$f_{\text{MAP}} = \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f) = \operatorname*{argmax}_{f \in \mathcal{F}} \ln p(\mathcal{D}|f) + \ln p(f) = \operatorname*{argmin}_{f \in \mathcal{F}} -\ln p(\mathcal{D}|f) - \ln p(f).$$

Example 16: Before we move on to finding the MAP solution, let's consider what the posterior can look like. Let us do this first by considering a discrete weight vector, with a prior and posterior that is a pmf. Again we use the coin example, where we are given a coin with an unknown probability w of seeing a 1 (a heads). The manufacturer that gave us the coin assures us that the bias is one of $\mathcal{F} = \{0.3, 0.5, 0.8\}$. Further, because it came from their factory, they know that the proportion of coins they produce are: 70% have $w = 0.3$, 20% have $w = 0.5$ and 10% have $w = 0.8$. In other words, the prior probability over outcomes for w is

$$p(w) = \begin{cases} 0.7 & w = 0.3 \\ 0.2 & w = 0.5 \\ 0.1 & w = 0.8 \end{cases}$$

Imagine we obtain a dataset of n coin flips, $\mathcal{D} = \{0, 1, 0, 1, 1, \ldots, 1\}$ namely where $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 1, \ldots, x_n = 1$. Given w, the distribution over an outcome is $p(x_i|w) = w^{x_i}(1-w)^{1-x_i}$. To compute the posterior $p(w|\mathcal{D})$, we need to compute the likelihood $p(\mathcal{D}|w)$ and $p(\mathcal{D})$; we already have the prior $p(w)$. Notice that the posterior $p(w|\mathcal{D})$ is a pmf with three probabilities: $p(w = 0.3|\mathcal{D})$, $p(w = 0.5|\mathcal{D})$ and $p(w = 0.8|\mathcal{D})$. To obtain the posterior we need to compute these three probabilities. Notice that

$$p(w = 0.3|\mathcal{D}) = \frac{p(\mathcal{D}|w = 0.3)p(w = 0.3)}{p(\mathcal{D})}$$

Let's start by computing $p(\mathcal{D}|w)p(w)$ for each w. Again recall that

$$p(\mathcal{D}|w) = \prod_{i=1}^{n} p(x_i|w) = \prod_{i=1}^{n} w^{x_i}(1-w)^{1-x_i} = w^{n_1}(1-w)^{n_0}$$

where n_1 is the number of observed 1s and n_0 the number of observed 0s, with $n = n_1 + n_0$.

As in Example 14, imagine that $n = 10$ and $n_1 = 8$ and $n_0 = 2$. Then we can evaluate $p(\mathcal{D}|w)p(w)$ for all three values of w:

$$\begin{aligned} p(\mathcal{D}|w = 0.3)p(w = 0.3) &= 0.3^{n_1}0.7^{n_0}0.7 \approx 2.25 \times 10^{-5} \\ p(\mathcal{D}|w = 0.5)p(w = 0.5) &= 0.5^{n_1}0.5^{n_0}0.2 \approx 0.0001953 \\ p(\mathcal{D}|w = 0.8)p(w = 0.8) &= 0.8^{n_1}0.2^{n_0}0.1 \approx 0.0006711 \end{aligned}$$

Now we can compute

$$\begin{aligned} p(\mathcal{D}) &= \sum_{w \in \mathcal{F}} p(\mathcal{D}, w) = \sum_{w \in \mathcal{F}} p(\mathcal{D}|w)p(w) \\ &\approx 2.25 \times 10^{-5} + 0.0001953 + 0.0006711 = 0.0008889 \end{aligned}$$

and finally get

$$p(w|\mathcal{D}) = \begin{cases} 2.25 \times 10^{-5}/p(\mathcal{D}) & w = 0.3 \\ 0.0001953/p(\mathcal{D}) & w = 0.5 \\ 0.0006711/p(\mathcal{D}) & w = 0.8 \end{cases} = \begin{cases} 0.0253 & w = 0.3 \\ 0.2197 & w = 0.5 \\ 0.7550 & w = 0.8 \end{cases}$$

With only 10 coin flips, even though the ratio of 1s to 0s suggested $w = 0.8$ is the bias, the posterior still has relatively high probability on $w = 0.5$, because the prior was higher for 0.5 than 0.8. If we had much more data, say $n_1 = 80$ and $n_0 = 20$, meaning $n = 100$, the posterior is much more concentrated. Going through the same steps as above, we would get

$$p(w|\mathcal{D}_{100}) = \begin{cases} 4.4 \times 10^{-23} & w = 0.3 \\ 85 \times 10^{-9} & w = 0.5 \\ 0.99999999 & w = 0.8 \end{cases}$$

This posterior effectively has all probability on $w = 0.8$, and we can be very confident that w is 0.8. Notice for either the dataset of 10 or 100 samples, the MAP estimate would actually be the same, since $p(w = 0.8|\mathcal{D})$ had the highest probability for both. □

In some situations we may not have a reason to prefer one model over another and can think of $p(f)$ as a constant over the model space $\mathcal{F}$. Namely, if $p(f) = c$ for some constant c, then MAP reduces to the maximization of the likelihood function

$$\operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)p(f) = \operatorname*{argmax}_{f \in \mathcal{F}} c \cdot p(\mathcal{D}|f) = \operatorname*{argmax}_{f \in \mathcal{F}} p(\mathcal{D}|f)$$

because the constant c comes out of the maximization and does not affect the relative ordering. This solution is the MLE solution described above, and further motivates why MLE is a sensible objective. Formally speaking, the assumption that $p(f)$ is constant is problematic because a uniform distribution cannot be always defined (say, over $\mathbb{R}$), though there are some solutions to this issue using improper priors. Nonetheless, it is useful conceptually to think of MLE as a special case of MAP estimation with a uniform prior.

Example 17: Let $\mathcal{D} = \{2, 5, 9, 5, 4, 8\}$ again be an i.i.d. sample from Poisson(λ_0), but now we are also given additional information. Suppose the prior knowledge about λ_0 can be expressed using a gamma distribution with parameters $k = 3$ and $\theta = 1$. Find the MAP estimate of λ_0.

First, we write the probability density function of the gamma distribution for our prior

$$p(\lambda) = \frac{\lambda^{k-1} e^{-\frac{\lambda}{\theta}}}{\theta^k \Gamma(k)},$$

where $\lambda > 0$. $\Gamma(k)$ is the gamma function that generalizes the factorial function; when k is an integer, we have $\Gamma(k) = (k-1)!$. The MAP estimate of the parameters can be found as

$$\lambda_{\text{MAP}} = \operatorname*{argmax}_{\lambda \in (0,\infty)} p(\mathcal{D}|\lambda)p(\lambda).$$

As before, we take the log to simplify calculations to get

$$\begin{aligned} \ln p(\mathcal{D}|\lambda)p(\lambda) &= \ln p(\mathcal{D}|\lambda) + \ln p(\lambda) \\ &= \sum_{i=1}^{n} \ln p(x_i|\lambda) + \ln p(\lambda). \end{aligned}$$

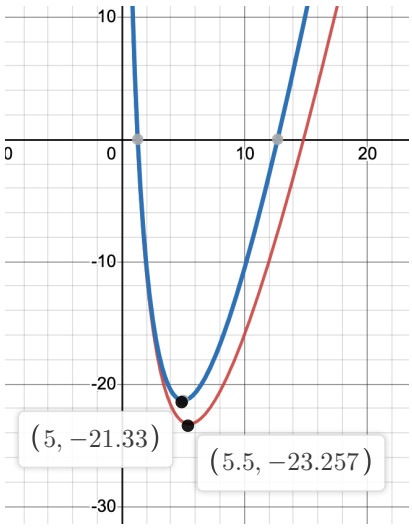


Figure 5.1: Contrasting the objective functions for the MLE (red) and MAP (blue) estimates for a Poisson distribution, in Examples 15 and 17. The MLE objective (red) has minimum at 5.5 and the MAP objective (blue) has minimum at 5. The objective functions are
$c(\lambda) = -33\ln(\lambda) + 6\lambda$
$c(\lambda) = -35\ln(\lambda) + 7\lambda$
where we have dropped constants. The objectives are not too different. The MAP objective simply balances the data likelihood with the prior, which is a gamma distribution with $k = 3$ and $\theta = 1$. The mode for this prior is $(k-1)\theta = 2$, and the mean is $k\theta = 3$, so the prior suggested that a smaller w (λ) was more likely. Consequently, the MAP solution is smaller than the MLE one.

We have already simplified the first term in the previous example. For the log of the prior distribution, we have

$$\begin{aligned}\ln p(\lambda) &= \ln\left(\lambda^{k-1}e^{-\frac{\lambda}{\theta}}\right) - \ln(\theta^k\Gamma(k)) \\ &= (k-1)\ln\lambda - \frac{\lambda}{\theta} - \ln(\theta^k\Gamma(k)).\end{aligned}$$

The last term is constant with respect to λ; so when we take the derivative it will disappear and we will be able to avoid computing it. Plugging everything back in

$$\ln p(\mathcal{D}|\lambda) + \ln p(\lambda) = \ln\lambda\sum_{i=1}^{n} x_i - n\lambda - \sum_{i=1}^{n}\ln(x_i!) + (k-1)\ln\lambda - \frac{\lambda}{\theta} - \ln(\theta^k\Gamma(k))$$

and taking the derivative gives

$$\frac{\partial \ln p(\mathcal{D}|\lambda) + \ln p(\lambda)}{\partial\lambda} = \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n + \frac{k-1}{\lambda} - \frac{1}{\theta} \qquad \text{because } \frac{\partial \ln p(\lambda)}{\partial\lambda} = \frac{k-1}{\lambda} - \frac{1}{\theta}$$

Once again setting the derivative to zero and solving for λ gives

$$\begin{aligned}\lambda_{\text{MAP}} &= \frac{k - 1 + \sum_{i=1}^{n} x_i}{n + \frac{1}{\theta}} \\ &= 5 \text{ for the dataset } \mathcal{D}\end{aligned}$$

□

A quick look at λ_{MAP} and λ_{MLE} suggests that as n grows, both numerators and denominators in the expressions above become increasingly more similar. In fact, it is a well-known result that, in the limit of infinite samples, both the MAP and MLE converge to the same model, f, as long as the prior does not have zero probability (or density) on f. This result shows that the MAP estimate approaches the MLE solution for large data sets. In other words, large data diminishes the importance of prior knowledge.

To get some intuition for this result, we will show that the MAP and MLE estimates converge to the same solution for the above example with a Poisson distribution. Let $s_n =$

$\sum_{i=1}^n x_i$, which is a sample from the random variable $S_n = \sum_{i=1}^n X_i$. If $\lim_{n\to\infty} s_n/n^2 = 0$ (i.e., s_n does not grow faster than n^2), then

$$\begin{aligned}
|\lambda_{\text{MAP}} - \lambda_{\text{MLE}}| &= \left| \frac{k-1+s_n}{n+1/\theta} - \frac{s_n}{n} \right| \\
&= \left| \frac{k-1}{n+1/\theta} - \frac{s_n}{n(n+1/\theta)} \right| \\
&\leq \frac{|k-1|}{n+1/\theta} + \frac{s_n}{n(n+1/\theta)} \qquad \xrightarrow[n\to\infty]{} 0
\end{aligned}$$

Note that if $\lim_{n\to\infty} s_n/n^2 \neq 0$, then both estimators go to ∞; however, such a sequence of values has an essentially zero probability of occurring. Consistency theorems for MLE and MAP estimation state that convergence to the true parameters occurs "almost surely" or "with probability 1" to indicate that these unbounded sequences constitute a set of measure-zero, under certain reasonable conditions (for more, see [11, Theorem 9.13]).

Example 18: Let $\mathcal{D} = \{x_i\}_{i=1}^n$ be an i.i.d. sample from a univariate Gaussian distribution. Our goal is to find the maximum likelihood estimates of the parameters. We start by forming the log-likelihood function

$$\begin{aligned}
\ln p(\mathcal{D}|\mu,\sigma) &= \ln \prod_{i=1}^n p(x_i|\mu,\sigma) \\
&= n \ln \frac{1}{\sqrt{2\pi}} + n \ln \frac{1}{\sigma} - \frac{\sum_{i=1}^n (x_i-\mu)^2}{2\sigma^2}.
\end{aligned}$$

We compute the partial derivatives of the log-likelihood with respect to all parameters as

$$\begin{aligned}
\frac{\partial}{\partial \mu} \ln p(\mathcal{D}|\mu,\sigma) &= \frac{\sum_{i=1}^n (x_i-\mu)}{\sigma^2} \\
\frac{\partial}{\partial \sigma} \ln p(\mathcal{D}|\mu,\sigma) &= -\frac{n}{\sigma} + \frac{\sum_{i=1}^n (x_i-\mu)^2}{\sigma^3}.
\end{aligned}$$

From here, we can solve for each variable that makes these equations zero, to derive that

$$\begin{aligned}
\mu_{\text{MLE}} &= \frac{1}{n} \sum_{i=1}^n x_i \\
\sigma^2_{\text{MLE}} &= \frac{1}{n} \sum_{i=1}^n (x_i - \mu_{\text{MLE}})^2.
\end{aligned}$$

Notice that the resulting σ^2_{MLE} is guaranteed to be non-negative, and so satisfies the conditions on that variable. □

MAP and MLE estimates are called *point estimates*. These estimates contrast Bayesian estimates, which estimate the entire posterior distribution for the parameters, as we discuss in the next section.

5.3 Bayesian Estimation

Maximum a posteriori and maximum likelihood approaches report the solution that corresponds to the mode of the posterior distribution and the likelihood function, respectively.

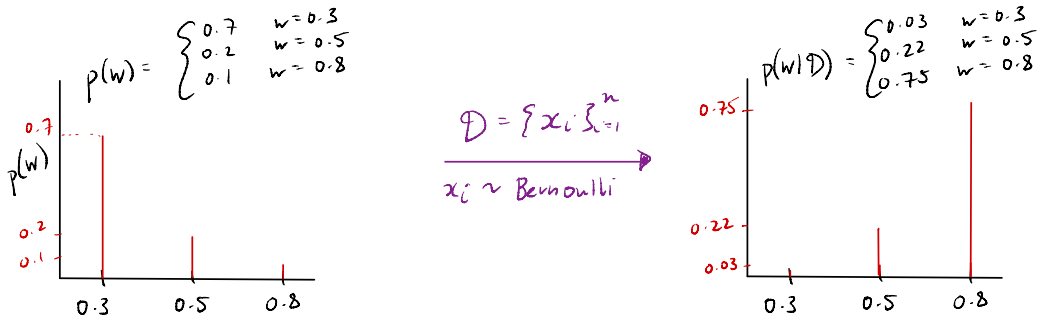


Figure 5.2: The prior and posterior for Example 16 about estimating the bias of a coin, where the bias of the coin is $w \in \{0.3, 0.5, 0.8\}$.

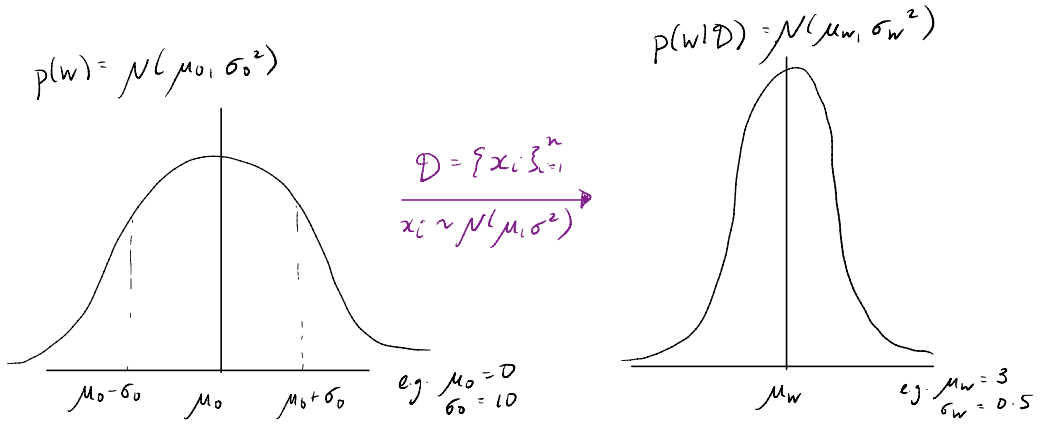


Figure 5.3: The prior and posterior for real-valued w, assuming that data is generated by a Gaussian. The prior is a wider distribution (larger variance), indicating that before seeing the data, we think it is most likely that w is near μ_0 (e.g., $\mu_0 = 0$), but are not confident about it. As we see more data, the mean μ_w shifts away from μ_0 towards a more plausible value given the data, and the variance shrinks. In the limit, as we see more and more samples, the variance shrinks to zero.

These approaches, however, do not consider the possibility of skewed distributions or multimodal distributions for the posterior distribution, nor do they allow us to reason about the distribution over plausible parameters. Bayesian estimation addresses those concerns.

5.3.1 Using the posterior

Bayesian approaches require estimation of the full posterior $p(w|\mathcal{D})$ (not just the mode as in MAP). We have already seen a case where we explicitly computed $p(w|\mathcal{D})$, for coins in Example 16 where $w \in \{0.3, 0.5, 0.8\}$. The posterior is computed by updating the prior with evidence (the data). We visualize this change for the coins example, in Figure 5.2. The weights w can also be real-valued, as they were when w was the mean of a distribution. The prior $p(w)$ might be a Gaussian distribution, and under certain conditions, the posterior $p(w|\mathcal{D})$ will also be a Gaussian, with a narrower variance, as visualized in Figure 5.3. We discuss how to actually get this $p(w|\mathcal{D})$ in the next subsection, but first let us consider how

we might use it.

One reason we want the posterior is to reason about the range of plausible parameters, given our data. For example, if $p(w|\mathcal{D})$ is Gaussian with mean μ and variance σ^2, then we can determine the interval around the mean μ that corresponds to 95% probability

$$p(w \in [\mu - \epsilon, \mu + \epsilon]) = 0.95 \implies \epsilon = 1.96\sigma.$$

This interval is called the *credible interval*, because any value for w in that region is plausible or credible. The MAP estimator would say that the most likely $w = \mu$, and the credible interval additionally provides some level of confidence in that estimate. If the variance is very small, then the credible interval is narrow and we can be reasonably confident that, under our probabilistic assumptions, we have seen enough data to identify w.[2]

On the other hand, if the variance of the posterior is high, then we are not that sure about our parameter. The credible interval indicates that values pretty far from μ are also plausible. Recall that the density at just one standard deviation away, at $\mu + \sigma$, remains quite high. This means that there is a wide range of plausible w, and it would be hard to confidently state that $w = \mu$ is near optimal.

We can also use this posterior to pick different point estimates. The MAP estimator uses the most likely point, the *mode* of the posterior. But, we could also use the mean or median of the posterior. For these notes, we will focus on the use of the credible interval, rather than on these alternative point estimates.[3]

5.3.2 Computing the posterior with conjugate priors

Now let us turn to how to obtain this posterior, so that we can actually extract these credible intervals. We have a formula for the posterior, because it can be expressed in terms of known distributions using Bayes rule

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}.$$

The difficulty, though, is in computing $p(\mathcal{D})$. For MAP, we did not need to estimate $p(\mathcal{D})$ because it was a constant that did not affect finding the most likely parameters. Now, we explicitly need to estimate this term, which we can do using

$$p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w)dw$$

Now computing the posterior $p(w|\mathcal{D})$ involves solving integrals. In some situations, these integrals can be solved analytically; in others, numerical integration is necessary. There are classes of distributions for which we know a simple form for the posterior. We discuss this below—with the concept of conjugate priors—but first give an example to get some intuition.

Example 19: Let $\mathcal{D} = \{2, 5, 9, 5, 4, 8\}$ yet again be an i.i.d. sample from Poisson(λ_0). Suppose the prior knowledge about the parameter of the distribution can be expressed

[2] Of course, it does not tell us anything about if we chose the wrong probabilistic assumptions. We will discuss this more, particularly discussing the non-realizable setting, in Section 10.3.

[3] For your own interest, to find more information about this topic, these alternative point estimates are called *Bayes estimators* and they are found by minimizing what is called the *posterior risk*.

using a gamma distribution with parameters $k = 3$ and $\theta = 1$. Let's find the posterior distribution and the resulting Bayesian estimate of λ_0, that is $\mathbb{E}[\Lambda|\mathcal{D}]$.

Let us first write the posterior distribution as

$$p(\lambda|\mathcal{D}) = \frac{p(\mathcal{D}|\lambda)p(\lambda)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\lambda)p(\lambda)}{\int_0^\infty p(\mathcal{D}|\lambda)p(\lambda)d\lambda},$$

where, as shown in previous examples, we have that

$$p(\mathcal{D}|\lambda) = \frac{\lambda^{\sum_{i=1}^n x_i} \cdot e^{-n\lambda}}{\prod_{i=1}^n x_i!} \qquad \text{and} \qquad p(\lambda) = \frac{\lambda^{k-1}e^{-\frac{\lambda}{\theta}}}{\theta^k \Gamma(k)}.$$

Before calculating $p(\mathcal{D})$, let us first note that

$$\int_0^\infty x^{\alpha-1}e^{-\beta x}dx = \frac{\Gamma(\alpha)}{\beta^\alpha}.$$

Now, we can derive that

$$\begin{aligned} p(\mathcal{D}) &= \int_0^\infty p(\mathcal{D}|\lambda)p(\lambda)d\lambda \\ &= \int_0^\infty \frac{\lambda^{\sum_{i=1}^n x_i} \cdot e^{-n\lambda}}{\prod_{i=1}^n x_i!} \cdot \frac{\lambda^{k-1}e^{-\frac{\lambda}{\theta}}}{\theta^k \Gamma(k)} d\lambda \\ &= \frac{\Gamma(k + \sum_{i=1}^n x_i)}{\theta^k \Gamma(k) \prod_{i=1}^n x_i! (n + \frac{1}{\theta})^{\sum_{i=1}^n x_i + k}} \end{aligned}$$

and subsequently that

$$\begin{aligned} p(\lambda|\mathcal{D}) &= \frac{p(\mathcal{D}|\lambda)p(\lambda)}{p(\mathcal{D})} \\ &= \frac{\lambda^{\sum_{i=1}^n x_i} \cdot e^{-n\lambda}}{\prod_{i=1}^n x_i!} \cdot \frac{\lambda^{k-1}e^{-\frac{\lambda}{\theta}}}{\theta^k \Gamma(k)} \cdot \frac{\theta^k \Gamma(k) \prod_{i=1}^n x_i! (n + \frac{1}{\theta})^{\sum_{i=1}^n x_i + k}}{\Gamma(k + \sum_{i=1}^n x_i)} \\ &= \frac{\lambda^{k-1+\sum_{i=1}^n x_i} \cdot e^{-\lambda(n+1/\theta)} \cdot (n + \frac{1}{\theta})^{\sum_{i=1}^n x_i + k}}{\Gamma(k + \sum_{i=1}^n x_i)}. \end{aligned}$$

Though this looks complex, notice that this is actually just a gamma distribution! The parameters k', θ' for this gamma distribution are

$$\begin{aligned} k' &= k + \sum_{i=1}^n x_i \\ \theta' &= \frac{\theta}{n\theta + 1} = \frac{1}{n + 1/\theta}. \end{aligned}$$

□

This example highlights why the selection of the prior distribution has important implications on calculation of the posterior mean. We did not pick the gamma distribution by chance: when the likelihood was multiplied by the prior, the resulting distribution remained in the same class of functions as the prior. Such prior distributions are called *conjugate*

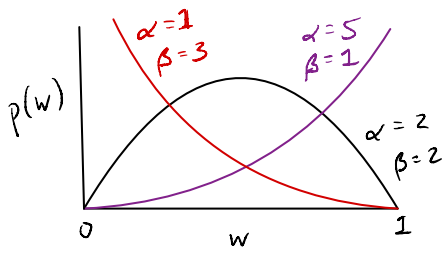


Figure 5.4: The beta distribution for different α, β. When $\alpha = \beta = 2$, the most likely value for w is 0.5, decreasing symmetrically away from 0.5. If we increase them, but keep them equal, such as $\alpha = \beta = 100$, we see similar behavior—the most likely value for w is 0.5, decreasing symmetrically away from 0.5—but the variance reduces and the distribution is more peaked around 0.5. For $\alpha = 5$ and $\beta = 1$, the mode is 1 and for $\alpha = 1$ and $\beta = 3$ the mode is 0.

priors. More formally, a prior is a conjugate prior to a likelihood if the resulting posterior is the same type of distribution as the prior (e.g, both are Gaussian distributions, or both are gamma distributions).

Why is it useful to have a conjugate prior? Consider if we did not have a conjugate prior. Imagine if instead we chose an exponential distribution as the prior for λ in our above example. When computing $p(\mathcal{D})$ and $p(\lambda|\mathcal{D})$, you would simply be stuck with complex integrals and formulas that do not correspond to any known distribution. One ramification is that it is not clear how to extract a credible interval. When you pick the conjugate prior—the gamma distribution—you can simply go look up the known formula for the parameters of the gamma posterior, assuming $p(x|\theta)$ is Poisson and prior $p(\theta)$ is gamma with parameters k and θ. You wouldn't even have to go through the above derivation; you would simply have been able to immediately know that $k' = k + \sum_{i=1}^{n} x_i$ and $\theta' = \frac{\theta}{n\theta+1}$.

Example 20: Let us return to the coins example, now allowing for the coin bias to be continuous, $w \in [0, 1]$. The likelihood $p(x|w)$ is a Bernoulli distribution for $x \in \{0, 1\}$, where $p(x|w) = w^x(1-w)^{(1-x)}$ or more simply $p(x = 1|w) = w$. The conjugate prior for the Bernoulli distribution is the *beta distribution*. The beta distribution has two positive parameters, $\alpha, \beta > 0$. The larger α is relative to β, the more the distribution is concentrated near 1, and vice versa. We visualize this distribution, with different α, β, in Figure 5.4.

Because the beta distribution is the conjugate prior to the Bernoulli distribution, we have a simple form for the posterior. After seeing a dataset $\mathcal{D} = \{x_i\}_{i=1}^n = \{1, 0, 0, \ldots, 1\}$ of coin flips, we update the prior distribution $p(w) = \text{Beta}(\alpha, \beta)$ to get posterior $p(w|\mathcal{D}) = \text{Beta}(\alpha + s_n, \beta + n - s_n)$ where $s_n = \sum_{i=1}^{n} x_i$ is the number of successes (flips that were 1). To get this posterior, we leveraged the known update for this conjugate prior.

The MAP estimate is the mode of this distribution. The mode of the beta distribution, for $\alpha, \beta > 1$, is $\frac{\alpha-1}{\alpha+\beta-2}$. Our posterior parameters are $\alpha' = \alpha + s_n$ and $\beta' = \beta + n - s_n$. As we get more and more samples, s_n dominates the small scalars α and β, and the mode approaches $\frac{s_n}{n}$, which is the proportion of times that we saw a 1 from the coin flip. For a smaller sample size, this counting estimate is skewed by the prior, towards the values deemed more plausible under the prior. For example, if $\alpha = 5$ and $\beta = 1$, then the prior puts much higher density near values of 1, with a mode of $\frac{\alpha-1}{\alpha+\beta-2} = \frac{5-1}{6-2} = 1$. Even if we see eight `0`s (tails) and only two `1` (heads), then our posterior is still somewhat skewed towards higher values, with a mode of

$$\frac{\alpha' - 1}{\alpha' + \beta' - 2} = \frac{5 + 2 - 1}{(5 + 2) + (1 + 8) - 2} = \frac{6}{14} \approx 0.429$$

In contrast to the MAP estimator, the MLE estimator would just use the proportion ob-

served in the data, and conclude that $w = 0.2$.

Finally, let use circle back to using this posterior to get a credible interval. We can compute intervals for many distributions, beyond the Gaussian. We typically assumed that the credible interval is centered, so that the probability above the interval equals $\delta/2$ and below the interval also equals $\delta/2$, with the remaining probability of $1 - \delta$ for the interval. You can obtain intervals for the beta distribution, using computing packages. For our example, if $\delta = 0.05$ to get a 95% credible interval, with $\alpha' = 7$ and $\beta' = 9$, the credible interval is $[0.21267, 0.67713]$. □

5.4 Maximum Likelihood for Conditional Distributions

We can also formulate MAP and maximum likelihood problems for conditional distributions. Recall that a conditional distribution has the form $p(y|x)$, for two random variables Y and X, where above we considered the marginal distribution $p(x)$ or $p(y)$. For the distributions above, we asked: what is the distribution over this variable? For a conditional distribution, we are instead asking: given some auxiliary information, now what is the distribution over this variable? When the auxiliary information changes, so will the distribution over the variable. For example, we may want to condition a distribution over sales of a particular product (Y) given the current month (X). We expect the distribution over Y to be different, depending on the month.

Conditional distributions can be from any of the distribution families discussed above, and we can similarly formulate parameter estimation problems. The parameters, however, are usually tied to the given variable X. We provide two simple examples to demonstrate this below. Much of the parameter estimation formulations we consider in the remainder of the book will be for conditional distributions, because in machine learning we typically have a large number of auxiliary variables (features) and are trying to predict (or learn the distribution over) targets. For regression and classification, we will see how many models can be formulated as maximum likelihood for conditional distributions $p(y|\mathbf{x})$.

Example 21: Assume you are given two random variables X and Y and that you believe $p(y|x) = \mathcal{N}(\mu = x, \sigma^2)$ for some unknown σ. Our goal is to estimate this unknown parameter σ. Notice that the distribution over Y varies, depending on which X value is observed.

We again start by forming the log-likelihood function, now for pairs of n samples $\mathcal{D} = (x_1, y_1), \ldots, (x_n, y_n)$. We will use the chain rule for probability: $p(x_i, y_i) = p(y_i|x_i)p(x_i)$.

$$\begin{aligned}
\ln p(\mathcal{D}|\sigma) &= \ln \prod_{i=1}^{n} p(x_i, y_i|\sigma) = \ln \prod_{i=1}^{n} p(y_i|x_i, \sigma)p(x_i) \\
&= \sum_{i=1}^{n} \ln p(y_i|x_i, \sigma) + \ln p(x_i) \\
&= \sum_{i=1}^{n} \ln \tfrac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - x_i)^2}{2\sigma^2}) + \ln p(x_i) \\
&= n \ln \frac{1}{\sqrt{2\pi}} + n \ln \frac{1}{\sigma} - \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{2\sigma^2} + \sum_{i=1}^{n} \ln p(x_i).
\end{aligned}$$

Notice that we use $\mu = x_i$ for each normal distribution $p(y_i|x_i, \sigma)$. We now compute the partial derivatives of the log-likelihood with respect to the parameter σ

$$\frac{\partial}{\partial \sigma} \ln p(\mathcal{D}|\sigma) = -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{\sigma^3}.$$

Notice that $\frac{\partial}{\partial \sigma} \sum_{i=1}^{n} \ln p(x_i) = 0$, because σ does not parameterize $p(x_i)$. Therefore, to obtain the optimal σ, we do not need to know or specify the distribution over the random variable X. By setting the derivative to zero, to obtain a stationary point, we obtain

$$\sigma^2_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2.$$

□

Exercise 14: We can also formalize this problem as a MAP problem. To do so, we need to pick a prior on the parameter σ. Let's pick a uniform distribution with a relatively narrow range of $[0.5, 2]$. Note that though MLE can be thought of as MAP with a uniform prior, this is only the case if we pick a uniform prior that does not restrict the space of feasible solution. This uniform prior, with range $[0.5, 2]$, is quite restrictive, and so we should get a different solution from above. Derive σ^2_{MAP}. □

The above example was chosen primarily for algebraic simplicity. More realistically, we might imagine that the mean of Y given x is a more general function of x. Let us revisit this example, assuming the $\mu = xw$ and that $\sigma = 1.0$. Then we have

$$\ln p(\mathcal{D}|w) = \sum_{i=1}^{n} \ln p(y_i|x_i, w) + \ln p(x_i) \implies \operatorname*{argmin}_{w \in \mathbb{R}} - \ln p(\mathcal{D}|w) = \operatorname*{argmin}_{w \in \mathbb{R}} - \sum_{i=1}^{n} \ln p(y_i|x_i, w)$$

where the equality comes from the fact the the optimization is the same when dropping the constant $\ln p(x_i)$. Further, we have

$$\begin{aligned} -\ln p(y_i|x_i, w) &= -\ln \tfrac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - x_i w)^2}{2\sigma^2}) && \triangleright\ \sigma = 1.0 \\ &= -\ln \tfrac{1}{\sqrt{2\pi}} + \tfrac{1}{2}(y_i - x_i w)^2 \end{aligned}$$

Again, dropping constants, namely $-\ln \frac{1}{\sqrt{2\pi}}$, we can define our objective as

$$c_i(w) = \tfrac{1}{2}(x_i w - y_i)^2 \qquad\qquad c(w) = \sum_{i=1}^{n} c_i(w)$$

where $\operatorname{argmin}_{w \in \mathbb{R}} c(w) = \operatorname{argmin}_{w \in \mathbb{R}} - \ln p(\mathcal{D}|w)$. Computing the gradient

$$\begin{aligned} \frac{\partial}{\partial w} c_i(w) = \frac{\partial}{\partial w} \tfrac{1}{2}(x_i w - y_i)^2 &= (x_i w - y_i) \frac{\partial}{\partial w} x_i w && \triangleright\ \text{using the chain rule} \\ &= (x_i w - y_i) x_i && \triangleright\ \text{using the derivative of } aw \text{ wrt } w \end{aligned}$$

Finally we can solve for w that makes the derivative zero.

$$0 = \frac{\partial}{\partial w} c(w) = \sum_{i=1}^{n} (x_i w - y_i) x_i = \sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} y_i x_i \qquad \implies w = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2} \tag{5.5}$$

This problem setting is actually called linear regression, where here we assumed the simpler setting of having one input x and one coefficient w. We will revisit the more general linear regression setting, in Chapter 12.

Example 22: Let us do another example where $p(y|x)$ is Gaussian, but this time let us consider a case where X is a discrete random variable. Imagine we are modeling profits Y from book sales, conditioned on whether the book is fiction or non-fiction. We let $\mathcal{X} = \{0, 1\}$ where $X = 0$ indicates that the book is a non-fiction book and $X = 1$ is that the book is a fiction book. Now we again have a different Gaussian for each input value, but since x only has two possible values, we only have two Gaussians. Formally, we have $p(y|X = 0) = \mathcal{N}(\mu_0, \sigma_0^2)$ and $p(y|X = 1) = \mathcal{N}(\mu_1, \sigma_1^2)$ for unknown parameters $\mathbf{w} = (\mu_0, \sigma_0, \mu_1, \sigma_1) \in \mathbb{R}^4$. We now have four parameters to learn.

We now need to formalize the MLE for this problem. To do so, again we simply need to derive the negative log likelihood for one pair (x_i, y_i), and the full MLE is the sum over all pairs. To make this simpler let us consider the two cases. Assume first that we have a sample (x_i, y_i) where $x_i = 0$, giving

$$-\ln p(y_i|x_i = 0, \mathbf{w}) = -\ln \tfrac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(y_i - \mu_0)^2}{2\sigma_0^2}\right). \tag{5.6}$$

We know how to find the MLE for a Gaussian, namely for μ_0, σ_0^2; we can exploit this here. But first, consider the other case, where $x_i = 1$, giving

$$-\ln p(y_i|x_i = 1, \mathbf{w}) = -\ln \tfrac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(y_i - \mu_1)^2}{2\sigma_1^2}\right). \tag{5.7}$$

We can write the negative log likelihood simply by choosing between one of these two formulas for each datapoint. If the datapoint has $x_i = 0$, then we use Equation (5.6). If the datapoint has $x_i = 1$, then we use Equation (5.7). Let $\mathcal{I}_0 \subset \{1, 2, \ldots, n\}$ be the set of indices where $x_i = 0$, namely $\mathcal{I}_0 = \{i \in \{1, 2, \ldots, n\} \mid x_i = 0\}$. Let $\mathcal{I}_1$ be the indices where $x_i = 1$. Notice these two sets are mutually exclusive, and cover the dataset because each x_i is either 0 or 1: $\mathcal{I}_0 \cup \mathcal{I}_1 = \{1, 2, \ldots, n\}$. We can write the negative log likelihood as

$$\sum_{i \in \mathcal{I}_0} -\ln p(y_i|x_i = 0, \mathbf{w}) + \sum_{i \in \mathcal{I}_1} -\ln p(y_i|x_i = 1, \mathbf{w}) = \sum_{i \in \mathcal{I}_0} -\ln p(y_i|\mu_0, \sigma_0) + \sum_{i \in \mathcal{I}_1} -\ln p(y_i|\mu_1, \sigma_1)$$

When solving for the MLE, we actually have two separate MLE problems: the first Gaussian for the datapoints where $x_i = 0$ and the second Gaussian for the datapoints where $x_i = 1$. We have already found the MLE solution for a Gaussian, which was the sample mean and (biased) sample variance. This means our MLE solution is

$$\mu_0 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} y_i \qquad \sigma_0^2 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} (y_i - \mu_0)^2$$

$$\mu_1 = \frac{1}{n_1} \sum_{i \in \mathcal{I}_1} y_i \qquad \sigma_1^2 = \frac{1}{n_1} \sum_{i \in \mathcal{I}_1} (y_i - \mu_1)^2$$

where $n_0 = |\mathcal{I}_0|$ is the number of points where $x_i = 0$ and $n_1 = |\mathcal{I}_1|$ the number of points where $x_i = 1$. This solution makes intuitive sense, since we are effectively modeling the Gaussian over profits for non-fiction books on the subset of the data about non-fiction books, and a completely separate Gaussian over profits for fiction books. □

5.5 Using Gradient Descent for Parameter Estimation

We found the stationary points in this section, using a closed-form solution. But we could have used gradient descent instead. For example, for the conditional model in the previous section, we could have initialize $w_0 = 0$ and updated using

$$w_{t+1} = w_t - \eta_t \sum_{i=1}^{n} (x_i w - y_i) x_i$$

This may not seem sensible, considering computing the closed-form solution is straightforward. For some models, however, we cannot obtain a closed-form solution. Further, as we discuss in the next chapter, it might actually be more efficient to use an iterative approach, namely with a modification to gradient descent called *stochastic gradient descent*. We discuss this optimization improvement first in the next chapter, before moving on to the more general prediction setting.

Exercise 15: Let us consider one example where we cannot obtain a closed-form solution. Imagine you decided it was unrealistic to assume that the variance is the same for all pairs (x, y).[4] You decide to parameterize the mean as xw_1 for $w_1 \in \mathbb{R}$ and the variance as $\exp(xw_2)$ for $w_2 \in \mathbb{R}$, where the exponential ensures the variance is positive. Write down the maximum likelihood problem, and derive the gradient descent update. □

[4] Uniform variance, though, is actually a relatively common assumption. It is called homoscedastic variance, whereas a variance that depends on the input x is called heteroscedastic.

Chapter 6

Stochastic Gradient Descent and Big Data Sets

Gradient descent provides a relatively generic approach to finding stationary points. The computational cost of gradient descent, however, can be quite high because we have to iterate over the entire dataset to compute the gradient. It is common to have very large datasets, in the millions of samples. Computing the gradient for each gradient descent step requires at least $\mathrm{O}(dn)$ where d is the size of the parameters and n is the number of samples. This is prohibitive for very large n.

One approach to handling big datasets is to use *stochastic approximation*, where the gradient is estimated using a stochastic sample. The idea is similar to why a sample average provides a reasonable estimate of the true mean. Instead of computing the gradient using all the samples, a random subsample provides a reasonably good approximation.

To see how this would be done, let us revisit the gradient of the objective function. Assume that we use a normalized objective, giving

$$c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w}) \qquad\qquad \nabla c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla c_i(\mathbf{w})$$

The full gradient is like computing the true expectation, where each point has uniform probability. In *stochastic gradient descent*, we use a small mini-batch of b samples (e.g., $b = 32$). For example, if using one sample to approximate the gradient ($b = 1$), you would randomly sample a datapoint i and update the weights on iteration t using

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla c_i(\mathbf{w}_t)$$

for some stepsize η_t.

At first glance, this might seem a little crazy. How can we get a good descent direction, with such a rough approximation to the gradient? The true gradient requires summing the gradients for all samples, and we use only one of those gradients! Can we even say that our algorithm will converge? Though this approach may appear to be too much of an approximation, there is a long theoretical and empirical history indicating its effectiveness (see for example [5, 4]). In fact, with modern dataset sizes that are very large, it is the most common strategy in use in machine learning.

The key idea for why this works is simple: the gradient for a single sample is an unbiased estimate of the true gradient. The true full gradient can be seen as an expectation with probability $p(i) = \frac{1}{n}$, where if K is a random index into $\{1, ..., n\}$

$$\mathbb{E}[\nabla c_K(\mathbf{w})] = \sum_{i=1}^{n} p(i) \nabla c_i(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla c_i(\mathbf{w}).$$

In stochastic gradient descent, we randomly sample K and so our update $\nabla c_K(\mathbf{w})$ is an unbiased estimate of $\frac{1}{n}\sum_{i=1}^{n}\nabla c_i(\mathbf{w})$. So, even if one step actually happens to be pointing in the wrong direction, on average across steps the gradient is pointing in the right direction and the weights will progressively move towards a stationary point.

We summarize this approach in Algorithm 2. Randomly sampling for each update would incur too much additional overhead. Instead, we can approximate this procedure by (a) first shuffling the order of points in the dataset and (b) then iterating in order over the entire dataset. Shuffling ensures that the order is randomized, and iterating over the entire dataset after shuffling once is more efficient that repeated random sampling. Only performing one iteration over the dataset, however, may not be enough. Instead, this procedure is repeated multiple times. Each iteration over the dataset is called an *epoch*.

Note that we give the concrete updates for linear regression as an example, which we introduce in a later chapter; we include them anyway for ease of reference when you need to implement mini-batch linear regression.

Algorithm 2: Stochastic Gradient Descent with $b = 1$ for $c(\mathbf{w}) = \frac{1}{n}c_i(\mathbf{w})$

Optimization parameters: number of epochs $= 10^4$
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
for $p = 1, \ldots$ number of epochs **do**
  Shuffle data points from $1, \ldots, n$
  for $k = 1, \ldots, n$ **do**
    $\mathbf{g} \leftarrow \nabla c_k(\mathbf{w})$ ▷ for linear regression, $\nabla c_k(\mathbf{w}) = (\mathbf{x}_k^\top \mathbf{w} - y_k)\mathbf{x}_k$
    // For convergence, the step-size η_t needs to decrease with time, such as $\eta_t = p^{-1}$
    // In practice, it is common to use stepsizes that do not decay with time
    // such as picking a small fixed stepsize ($\eta = 0.01$), or using stepsize adaptation
    $\eta \leftarrow p^{-1}$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta\mathbf{g}$
return $\mathbf{w}$

Setting $b = 1$ is typically too high-variance. Instead, it is more common to pick $b > 1$. The size of b is like the number of samples in a sample average: the larger the b, the lower the variance. However, increasing b also increases computation. So, we have to strike a balance. Notice that setting $b = n$ corresponds to a full batch gradient update. Using mini-batches allows us to select a b between the two extremes of $b = 1$ (high-variance, computationally cheap) and $b = n$ (zero-variance, computationally expensive). As is typically the case, the extremes are not the best choices and an interim value of b is preferred.

Exercise 16: Show that the mini-batch stochastic gradient is also an unbiased estimate of the batch gradient:

$$\frac{1}{b}\sum_{k=1}^{b}\nabla c_{i_k}(\mathbf{w}) \quad \text{for } i_1, \ldots, i_b \text{ sampled uniformly from } \{1, \ldots, n\}. \tag{6.1}$$

□

The mini-batch stochastic gradient descent approach is shown in Algorithm 3. Each step corresponds to updating with a randomly chosen mini-batch of data: a small subset of

size b from the larger dataset of size n. As with $b = 1$, we want to avoid randomly sampling a mini-batch for each update, as this would incur unnecessary overhead. Again, we shuffle the dataset and grab each mini-batch in-order over the entire dataset. For a dataset with (possibly paired) samples $z_1, z_2, \ldots, z_n$, that has been randomly shuffled to give new order $z_{i_1}, z_{i_2}, \ldots, z_{i_n}$, the mini-batches correspond to

$$\underbrace{|z_{i_1}, z_{i_2}, \ldots, z_{i_b}|}_{\text{1st mini-batch}} \underbrace{|z_{i_{b+1}}, z_{i_{b+1}}, \ldots, z_{i_{2b}}|}_{\text{2nd mini-batch}} \ldots \underbrace{|z_{i_{n-b+1}}, z_{i_{n-b}}, \ldots, z_{i_n}|}_{n/b \text{ mini-batch}}$$

where we assumed n is divisible by b. If it is not, then the last batch is simply smaller, consisting of the remaining points. In this case, the number of batches is $\lfloor \frac{n}{b} \rfloor + 1$.

Exercise 17: When we say that the dataset is shuffled, we only mean the ordering of the points. You cannot shuffle the inputs and target. In other words, we swap the order of $(\mathbf{x}_{i_1}, y_{i_1})$ and $(\mathbf{x}_{i_2}, y_{i_2})$ but we do not swap their targets. In fact, it would be really bad if we did! Why would it be bad to swap their targets? □

Remark: In many cases, the dataset is actually constantly growing. Data is constantly being gathered, and new samples are streaming in. In this streaming setting—also called an online setting—stochastic gradient descent allows us to avoid storing all the data, since we simply do updates with the most recent samples. If we assume that each objective is defined on an input pair $(\boldsymbol{X}, Y)$, writing $c(\mathbf{w}; (\boldsymbol{X}, Y))$, then the stochastic gradient update is an unbiased estimate of the gradient of the true expected error, $\mathbb{E}[c(\mathbf{w}; (\boldsymbol{X}, Y))]$ where the expectation is over all possible $(\boldsymbol{X}, Y)$. Each new datapoint is a random sample from the joint distribution over $(\boldsymbol{X}, Y)$.

Remark: The computational trade-off when increasing b is nuanced, because computing gradients is easily parallelizable. If you can parallelize across 32 cores, then there is almost no disadvantage to using $b = 32$ instead of $b = 1$ in terms of computation time. If you can parallelize across 256 cores, then even better! Of course, more computing energy is still being expended, and so even in these settings, the trade-off might be taken into account. There is a huge improvement when moving from $b = 1$ to $b = 8$, still potentially a large gain when moving to $b = 32$, but at some point we reach diminishing returns.

6.1 Stepsize Selection for SGD

SGD requires a new mechanism to pick stepsizes. The conditions for convergence of SGD include conditions on the step-sizes, requiring them to decrease over time. One simple choice is to set the stepsize to decay with the epoch number: $\eta = p^{-1}$. Smarter stepsize algorithms use statistics on the magnitude of the gradient. For example, similarly to an algorithm called AdaGrad [7], we can normalize the stepsize by the sum of accumulating gradients

$$\eta_t = \frac{1}{\sqrt{1 + \bar{g}_t}} = (1 + \bar{g}_t)^{-1/2} \tag{6.2}$$

where $\bar{g}_t = \bar{g}_{t-1} + \frac{1}{d+1} \sum_{j=0}^{d} g_{t,j}^2 = \bar{g}_{t-1} + \frac{1}{d+1} \|\mathbf{g}_t\|_2^2$. After many epochs, these stochastic gradient descent updates will converge, and oscillate around the true weight vector, with the decreasing step-size progressively smoothing out these oscillations.

An even smarter stepsize strategy is to use a different stepsize per dimension. The idea is that you might need to take a bigger step in one dimension and a smaller in another

Algorithm 3: SGD for objective $c(\mathbf{w}) = \frac{1}{n}c_i(\mathbf{w})$ with AdaGrad

Fix iteration parameters: number of epochs $= 10^4$ and mini-batch size $b = 32$
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
$\bar{\mathbf{g}} \leftarrow$ zero vector in $\mathbb{R}^d$
for $p = 1, \ldots$ number of epochs **do**
  Shuffle ordering of data points from $1, \ldots, n$
  for $k = 0, \ldots, \lfloor \frac{n}{b} \rfloor$ **do**
    $\mathbf{g} \leftarrow 0$
    $c \leftarrow 0$
    for $i = kb, \ldots, \min((k+1)b - 1, n)$ **do**
      $\mathbf{g} \leftarrow \mathbf{g} + \nabla c_i(\mathbf{w})$ ▷ for linear regression, $\nabla c_i(\mathbf{w}) = (\mathbf{x}_i^\top \mathbf{w} - y_i)\mathbf{x}_i$
      $c \leftarrow c + 1$
    $\mathbf{g} \leftarrow \mathbf{g}/c$ ▷ element-wise division
    for $j = 0, \ldots, d - 1$ **do**
      $\bar{\mathbf{g}}[j] \leftarrow \bar{\mathbf{g}}[j] + \mathbf{g}[j]^2$
      $\eta \leftarrow 1/(\sqrt{\bar{\mathbf{g}}[j] + 1})$
      $\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \eta \mathbf{g}[j]$
return $\mathbf{w}$

dimension. For example, if in one direction, the optimization surface is flatter, you might need a bigger stepsize, and if another it is steep, then you need a small stepsize. Most stepsize selection strategies select vectors of stepsizes, where each element in the vector corresponds to a stepsize for the corresponding dimension. In fact, this is how AdaGrad is designed. Using the same notation as above, we now have a vector $\boldsymbol{\eta}_t \in \mathbb{R}^{d+1}$ where

$$\boldsymbol{\eta}_t = (1 + \bar{\mathbf{g}}_t)^{-1/2} \tag{6.3}$$

where $\bar{\mathbf{g}}_t = \bar{\mathbf{g}}_{t-1} + \mathbf{g}_t^2$ using elementwise addition and powers. In other words, for each entry $\eta_{t,j}$ in the vector $\boldsymbol{\eta}_t$ and entry $\bar{g}_{t,j}$ in the vector $\bar{\mathbf{g}}_t$, we update $\bar{g}_{t,j} = \bar{g}_{t-1,j} + g_{t,j}^2$ and $\eta_{t,j} = (1 + \bar{g}_{t,j})^{-1/2}$. Then each entry in the weights is updated using $w_{t+1,j} = w_{t,j} - \eta_{t,j} g_{t,j}$. This is the stepsize approach we use in Algorithm 3.

Exercise 18: How does Algorithm 3 change if we want to use a constant stepsize, one based on the epoch number or the scalar stepsize heuristic above? Can you think of anyways to improve on the AdaGrad stepsize approach currently in Algorithm 3? □

6.2 Contrasting Computational Complexity of GD and SGD

We motivated the use of SGD for big datasets. If it costs[1] $\mathrm{O}(d)$ to compute $\nabla c_i(\mathbf{w})$, then computing each mini-batch update costs $\mathrm{O}(bd)$. At first glance, this seems much better than the cost of computing one full gradient update, which is $\mathrm{O}(nd)$. The missing factor

[1]This big-O notation means how the computation scales with the given variable. It omits constants, to focus on the key terms. For example, a methods that uses $3d$ computation and one that uses d computation both have $O(d)$ computational complexity. They both scale much better with d than an algorithm that requires $O(d^2)$ computation.

is the *number of updates* k that each algorithm performs. We expect the full gradient to be a much less noisy step, and so require much fewer iterations k to reach a stationary point. Each update, with an appropriately chosen stepsize, is guaranteed to improve the weights and reach a point with a lower objective value c. SGD with $b = 1$, on the other hand, is quite noisy; an approximate gradient may even point in the wrong direction! For this reason, we cannot guarantee improvement on each update, but rather only aggregate improvement across updates.

We are faced with the question: is $k_{\text{sgd}}bd$ actually smaller than $k_{\text{gd}}nd$? The answer depends on n and the level of noise in the problem, which affects k_{sgd}. First, we can see that for SGD to be preferred we need

$$k_{\text{sgd}}bd < k_{\text{gd}}nd \implies k_{\text{sgd}} < k_{\text{gd}}\frac{n}{b}$$

If $n = 1$ million and $b = 32$, then SGD would have to use 30,000 times more updates than GD for it to be worth using GD instead of SGD. Namely, even if GD used only 10 updates, as long as SGD uses fewer than 300,000 updates it is the preferable algorithm.

In general, we expect SGD to be better than full-batch GD. We can see GD as an extreme of SGD, where we set the batch size b to n. The two extremes are unlikely to be the best for SGD: $b = 1$ will likely have gradients that are too noisy and $b = n$ is likely wasting too much computation to reduce variance. An interim value of b is likely better than either of these two extremes. This is particularly true if we have a large n. If we do happen to have small number of samples (e.g., $n = 100$), then it is likely worth simply using GD. Once we start getting to more realistic sizes, even in the thousands, $b < n$ is likely better.

Chapter 7

Introduction to Prediction Problems

Machine learning addresses many problem settings, which can sometimes feel overwhelming. As a non-exhaustive list, these include supervised learning (with classification and regression); semi-supervised learning; unsupervised learning; completion under missing features; structured prediction; learning to rank; statistical relational learning; active learning; and temporal prediction (with time series prediction and policy evaluation in reinforcement learning and online learning). For some of these settings, such as active learning and reinforcement learning, the data collection is a central part of the algorithm and can significantly determine the quality of the learned predictive models. Most other settings assume that data has been collected—without our ability to influence that collection—and now we simply need to analyze that data and learn the best predictors that we can. In this passive setting, we can either assume that the data is i.i.d.—which is the most common—or that there are dependencies between data points—such as in time series prediction or statistical relational learning. There are also settings where the data is incomplete, say because a user did not fill in their age.

One ontology, therefore, could consider the following dimensions to categorize machine learning problems:

1. passive vs. active
2. i.i.d. vs. non-i.i.d.
3. complete vs. incomplete.

As with all ontologies, each problem will not perfectly fit into these categories. Further, it is likely that most data collection is not completely passive (even if only because the human modeler influences collection of data), is likely not i.i.d. (even if we intended it to be), and likely has some missing components. Nonetheless, algorithms will make these assumptions, to varying degrees, even if the data does not satisfy those assumptions. For these notes, we will focus on the simplest setting: passive, i.i.d. and complete.

In this chapter, we will first introduce classification and regression and then discuss criteria for selecting functions for classification and regression, to motivate the algorithms developed in later chapters.

7.1 Supervised Learning Problems

We start by defining a data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathcal{X}$ is the i-th *input* or *observation* and $y_i \in \mathcal{Y}$ the corresponding target. We usually assume that $\mathcal{X} = \mathbb{R}^d$, in which case $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ is a d-dimensional vector called an *instance*

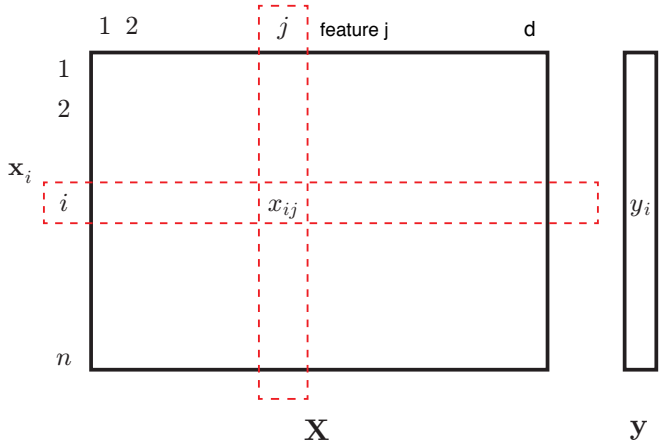


Figure 7.1: Notation for the dataset. $\mathbf{X}$ *is an* n*-by-*d *matrix, with rows corresponding to instances and columns to features.* y *is an* n*-by-*1 *vector of targets.*

or a *sample*.[1] Each dimension of x_j is typically called a *feature* or an *attribute*. We will often organize the dataset into a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ where each row corresponds to a sample $\mathbf{x}_i$ and each column corresponds to a feature (see Figure 7.1).

The distinction between $\mathbf{x}$ and y is due to the fact that we assume that the features are relatively easy to collect for each object (e.g., by measuring the height of a person or the square footage of a house), while the target variable is difficult to observe or expensive to collect (e.g., presence of a disease or the final selling price of a house before it has sold). Such situations usually benefit from the construction of a computational model that predicts targets from a set of input values. The model is trained using a set of input observations for which target values have already been collected. In deployment, we can use this model to make predictions from easy-to-obtain information—the observation—about hard-to-obtain information—the targets.

7.1.1 Regression and Classification

The differences in algorithms for prediction problems, with i.i.d. complete data, typically arises from the properties of the inputs (observations) and the properties of the targets. For example, we may need to treat text observations—such as those from a set of documents—differently than a ten-dimensional real-valued observation vector of sensor readings reflecting the temperature and pressure in a physical system. A simple, and relatively common strategy, to handle these differences is to map different types of observations—language, categorical variables and even sequence data—into a Euclidean space where the observation is re-represented as a real-valued vector. Many prediction algorithms are designed for real-valued observations, and so standard algorithms can then be applied. This question of data representation is a central problem in machine learning. For this course, we will assume the observations are already in a convenient form, as a d-dimensional real-valued vector.

The properties of the target are also important, and result in two typical distinctions for

[1] In statistics, a sample usually refers to a collection of randomly sampled $\mathbf{x}$, rather than a single instance. It is common in machine learning, though, to use the word sample to mean a single sample, rather than multiple samples or draws from the distribution.

prediction problems: classification and regression. Generally speaking, we have a regression problem when $\mathcal{Y}$ is continuous and a classification problem if $\mathcal{Y}$ is discrete. In **regression** possible target sets include $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [0, \infty)$. An example of a regression problem is shown in Table 7.1.

	size [sqft]	age [yr]	dist [mi]	inc [$]	dens [ppl/mi^2]	y
$\mathbf{x}_1$	1250	5	2.85	56,650	12.5	2.35
$\mathbf{x}_2$	3200	9	8.21	245,800	3.1	3.95
$\mathbf{x}_3$	825	12	0.34	61,050	112.5	5.10

Table 7.1: An example of a regression problem: prediction of the price of a house in a particular region. Here, features indicate the size of the house (size) in square feet, the age of the house (age) in years, the distance from the city center (dist) in miles, the average income in a one square mile radius (inc), and the population density in the same area (dens). The target indicates the price a house is sold at, e.g. in hundreds of thousands of dollars.

In **classification** we construct a function that predicts discrete class labels; this function is typically called a *classifier*. The cardinality of $\mathcal{Y}$ in classification problems is usually small, e.g. $\mathcal{Y} = \{\text{healthy}, \text{diseased}\}$. An example of a data set for classification with $n = 3$ data points and $d = 5$ features is shown in Table 7.2.

Classification problems can be further subdivided into multi-class and multi-label problems. A multi-class problem consists of providing the single label for an input. For example, for (simple) blood-type with $\mathcal{Y} = \{\text{A}, \text{B}, \text{AB}, \text{O}\}$, a patient can only be labeled with one of these labels. Within multi-class problems, if there are only two classes, it is called binary classification, such as the example in Table 7.2. In multi-label, an input can be associated with more than one label. An example of a multi-label problem is the classification of text documents into categories such as $\{\text{sports}, \text{medicine}, \text{travel}, \text{politics}\}$. Here, a single document may be related to more than one value in the set; e.g. an article on sports medicine. The learned function can now return multiple outputs.

Typically, to make the outputs more consistent between these two settings, the output for both multi-class and multi-label is an indicator vector. For $m = |\mathcal{Y}|$, the prediction for blood types might be [0 1 0 0] to indicate blood-type B and the prediction for four article labels could be [1 1 0 0] if it is both an article pertaining to sports and medicine.

	wt [kg]	ht [m]	T [°C]	sbp [mmHg]	dbp [mmHg]	y
$\mathbf{x}_1$	91	1.85	36.6	121	75	-1
$\mathbf{x}_2$	75	1.80	37.4	128	85	$+1$
$\mathbf{x}_3$	54	1.56	36.6	110	62	-1

Table 7.2: An example of a binary classification problem: prediction of a disease state for a patient. Here, features indicate weight (wt), height (ht), temperature (T), systolic blood pressure (sbp), and diastolic blood pressure (dbp). The class labels indicate presence of a particular disease, e.g. diabetes. This data set contains one positive data point ($\mathbf{x}_2$) and two negative data points ($\mathbf{x}_1$, $\mathbf{x}_3$). The class label shows a disease state, i.e. $y_i = +1$ indicates the presence while $y_i = -1$ indicates absence of disease.

A more precise distinction between regression and classification is that regression considers the *order* in the target variables, whereas classification assigns labels (unordered). For example, we might have the discrete set of prices for an item $\{1, 2, 3, 4, 5, \ldots, 100\}$ where the order matters. This problem is called *ordinal regression*. When considering accuracy of our prediction, we would want to say a prediction of 99 for a true target of 100 is closer than a prediction of 1. For classification, it is common to check accuracy by measuring if labeled the item correctly or not: here both 1 and 99 would be equally treated as incorrect labels. In other instances, a discrete set of labels $\{1, 2, 3, 4, 5\}$ may not actually be ordered: each number might just map to a property, like an inventory category. In that case, checking accuracy based on similarity would be incorrect. We will see this primary distinction between regression and classification in Section 7.2, when we formalize prediction accuracy.

In summary, a key distinction between supervised learning problems is whether the target variables are ordered or unordered. Regression problems are those where we want to exploit the ordering in the target variables, and classification problems are those where we treat target variables as unordered labels. It is also common to less precisely define regression as handling continuous targets, and classification as handling (finite) discrete targets. This somewhat more vague definition is common because many datasets either have continuous targets or a discrete set of labels, with ordinal targets less commonly considered. In these notes, we will only consider these two more common cases: real-valued targets and small discrete sets.

Exercise 19: Is the set {`Prefers apples, Prefers oranges, Prefers bananas`} ordered or unordered? How about {`Good, Better, Best`}? □

7.1.2 Deciding how to formalize the problem

We repeatedly return to the question: How do you decide which problem formulation to use? Though the mathematical procedures in machine learning are precise, deciding how to formulate real-world problems is subtle, and so inherently less clear-cut. The selection of a particular way of modeling depends on the analyst and their knowledge of the domain as well as technical aspects of learning.

We have already seen this with MLE and MAP, where an important step was to decide which distribution to use to model our data. For example, to model a pmf $p(x)$, we might choose to use a more restricted pmf like the Poisson distribution, because it only requires us to learn on parameter λ. If we use a table of probabilities—a categorical distribution—then we have to learn $k-1$ probabilities, if $x \in \{1, 2, \ldots, k\}$. If we have lots of data, then learning the more powerful distribution might be suitable. If we have very little data, then we might prefer to learn the more restricted pmf.

This same problem arises in supervised learning, because (as we will see) we are modeling $p(y|\mathbf{x})$. The biggest question for these models is what types of (nonlinear) functions we learn on $\mathbf{x}$, to best model this distribution. But, when deciding between using ordering or not, we are also asking what type of distribution to use for $p(y|\mathbf{x})$. For example, consider the output space $\mathcal{Y} = \{0, 1, 2\}$. We can treat this as a multi-class classification problem, or we could presume $\mathcal{Y} = [0, 2]$ and learn a regression model. We can then threshold the predictions returned by the regression model, by rounding them to the closest integer. We might choose to do so because there is an ordering to these variables and because regression functions can be easier to learn and often produce surprisingly good classification predictions.

The primary point of these examples is that formalizing the problem—selecting the function class, distributions and/or objective—does not have one clear-cut answer. But it is a critical step in using machine learning effectively. Fortunately, there is a wealth of knowledge, especially empirically, that can guide this selection. As you learn more about the methods, combined with some information about structure in your domain, you will become better at this specification. Nonetheless, picking distributions, function classes and objectives can be difficult, and is an art. Deciding to use regression and classification is actually typically more clear-cut, since we will almost always use the criteria that we use classification for unordered targets and regression for ordered targets.

7.2 Optimal Classification and Regression Models

Our goal is to establish the criteria that will be used to evaluate predictors $f : \mathcal{X} \to \mathcal{Y}$ and subsequently define optimal classification and regression models. To do so, we assume we have access to the true joint distribution $p(\mathbf{x}, y)$ and ask what the optimal prediction would be in this ideal case. The optimal predictor is defined based on a cost function $\text{cost} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, where $\text{cost}(\hat{y}, y)$ reflects the cost or penalty for predicting $\hat{y}$ when the true target is y. Because X, Y are random, the cost $C = \text{cost}(f(X), Y)$ is also a random variable, because it is a function of these random variables. Our goal is to minimize the expected cost. We first consider examples of costs, and then derive the optimal predictors.

7.2.1 Examples of costs

The costs for classification and regression are usually different. A typical cost function for classification is the *0-1 cost*,

$$\text{cost}(\hat{y}, y) = \begin{cases} 0 & \text{when } y = \hat{y} \\ \\ 1 & \text{when } y \neq \hat{y} \end{cases} \tag{7.1}$$

Notice that this equally applies to binary classification problems or multi-class classification problems. It simply reflects: did you get the class prediction right or wrong.

		Y	
		-1 (¬Has Disease)	1 (Has Disease)
$\hat{Y}$	-1 (¬Has Disease, No Test)	0	1000
	1 (Has Disease, Do Test)	1	1

Table 7.3: The cost function for the medical lab, cost$(\hat{y}, y)$, with $c_{\text{lawsuit}} = 1000$ and $c_{\text{lab}} = 1$.

A more complex cost function might arise in settings where certain inaccurate predictions are more problematic than others. Let's consider a concrete example, in a medical domain. Suppose our goal is to decide whether a patient with a particular set of symptoms ($\mathbf{x}$) should be sent for an additional lab test ($y = 1$ if yes and $y = -1$ if not), with cost c_{lab}, in order to improve diagnosis. However, if we do not perform a lab test and the patient is later found to have needed the test for proper treatment, we may incur a significant penalty, say

c_{lawsuit}. If $c_{\text{lawsuit}} \gg c_{\text{lab}}$, as it is expected to be, then the classifier needs to appropriately adjust its outputs to account for the cost disparity in different forms of incorrect prediction. Here, the cost is better depicted as a table, in Table 7.3. If there is no such asymmetry in your problem, where false negatives are more costly than false positives (and vice-versa), then a reasonable default is the 0-1 cost in Equation (7.1).

In regression, common costs are the squared error

$$\text{cost}(\hat{y}, y) = (\hat{y} - y)^2 \tag{7.2}$$

and the absolute error

$$\text{cost}(\hat{y}, y) = |\hat{y} - y|. \tag{7.3}$$

The squared error more heavily penalizes values further away from y than the absolute error. There are many other costs, that factor in the magnitude of the targets, such as the percentage error.

7.2.2 Deriving the optimal predictors

We begin first by deriving the optimal classifier. We can express the expected cost as follows, assuming the inputs are continuous real-valued vectors and the targets are from a discrete set $\mathcal{Y}$ and $\hat{y} = f(\mathbf{x})$ for the given predictor f

$$\begin{aligned}\mathbb{E}[C] &= \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} \text{cost}(f(\mathbf{x}), y) p(\mathbf{x}, y) d\mathbf{x} \\ &= \int_{\mathcal{X}} p(\mathbf{x}) \sum_{y \in \mathcal{Y}} \text{cost}(f(\mathbf{x}), y) p(y|\mathbf{x}) d\mathbf{x},\end{aligned}$$

where the integration is over the entire input space $\mathcal{X} = \mathbb{R}^d$. Notice that we have to predict one class for each observation: $f(\mathbf{x})$ can only output one value $\hat{y}$ in $\mathcal{Y}$. But, the target is random. Because of this the optimal classifier f^* may not be able to obtain zero cost. However, simply by looking at the above equation, we can obtain $f^* = \text{argmin}\, \mathbb{E}[C]$, by picking the best classifier for each $\mathbf{x}$ separately

$$\begin{aligned}f^*(\mathbf{x}) &= \underset{\hat{y} \in \mathcal{Y}}{\text{argmin}}\, \mathbb{E}[C|X = \mathbf{x}] \\ &= \underset{\hat{y} \in \mathcal{Y}}{\text{argmin}} \sum_{y \in \mathcal{Y}} \text{cost}(\hat{y}, y) p(y|\mathbf{x}).\end{aligned}$$

This classifier is called the *Bayes risk classifier*.

If we use the 0-1 cost function, in Equation (7.1), the Bayes risk classifier simply becomes

$$
\begin{aligned}
f^*(\mathbf{x}) &= \underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmin}} \sum_{y\in\mathcal{Y}} \operatorname{cost}(\hat{y}, y) p(y|\mathbf{x}) \\
&= \underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmax}} \left(1 - \sum_{y\in\mathcal{Y}} \operatorname{cost}(\hat{y}, y) p(y|\mathbf{x})\right) \\
&= \underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmax}} \sum_{y\in\mathcal{Y}} (1 - \operatorname{cost}(\hat{y}, y)) p(y|\mathbf{x}) && \triangleright \text{ because } \sum_{y\in\mathcal{Y}} p(y|\mathbf{x}) = 1 \\
&= \underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmax}} \sum_{y\in\mathcal{Y}, y\neq\hat{y}} 0 \cdot p(y|\mathbf{x}) + \sum_{y\in\mathcal{Y}, y=\hat{y}} 1 \cdot p(y|\mathbf{x}) \\
&= \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, p(y|\mathbf{x})
\end{aligned}
$$

Therefore, if $p(y|\mathbf{x})$ is known or can be accurately learned, we are fully equipped to make the prediction that minimizes the total cost. In other words, we have converted the problem of minimizing the expected classification cost or probability of error, into the problem of learning functions, more specifically learning probability distributions.

The analysis for regression is similar to that for classification. Here too, we are interested in minimizing the expected cost of prediction of the true target y when a predictor $f(\mathbf{x})$ is used. The expected cost can be expressed as

$$\mathbb{E}[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} \operatorname{cost}(f(\mathbf{x}), y) p(\mathbf{x}, y) dy d\mathbf{x}.$$

For simplicity, we will consider the squared error from Equation (8.1)

$$\operatorname{cost}(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2,$$

which results in

$$
\begin{aligned}
\mathbb{E}[C] &= \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(\mathbf{x}) - y)^2 p(\mathbf{x}, y) dy d\mathbf{x} \\
&= \int_{\mathcal{X}} p(\mathbf{x}) \underbrace{\int_{\mathcal{Y}} (f(\mathbf{x}) - y)^2 p(y|\mathbf{x}) dy}_{g(f(\mathbf{x}))} d\mathbf{x}.
\end{aligned}
$$

Assuming $f(\mathbf{x})$ is flexible enough to be separately optimized for each unit volume $d\mathbf{x}$, we see that minimizing $\mathbb{E}[C]$ leads us to the problem of finding $\hat{y}$ for each $\mathbf{x}$ to minimize

$$g(\hat{y}) = \int_{\mathcal{Y}} (\hat{y} - y)^2 p(y|\mathbf{x}) dy.$$

To find the optimal $\hat{y}$, we can solve this minimization problem by finding a stationary point, the global minimum. To do so, we differentiate g with respect to $\hat{y}$ and find the point where

the derivative equals zero

$$\begin{aligned}
&\frac{\partial g(\hat{y})}{\partial \hat{y}} = 2\int_{\mathcal{Y}} (\hat{y} - y)p(y|\mathbf{x})dy = 0 \\
&\implies \hat{y}\underbrace{\int_{\mathcal{Y}} p(y|\mathbf{x})dy}_{=1} = \int_{\mathcal{Y}} yp(y|\mathbf{x})dy \\
&\implies \hat{y}\underbrace{\int_{\mathcal{Y}} p(y|\mathbf{x})dy}_{=1} = \int_{\mathcal{Y}} yp(y|\mathbf{x})dy \\
&\implies \hat{y} = \int_{\mathcal{Y}} yp(y|\mathbf{x})dy = \mathbb{E}[Y|\mathbf{x}].
\end{aligned}$$

Therefore, the optimal predictor is

$$f^*(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}].$$

Therefore, the optimal regression model in the sense of minimizing the square error between the prediction and the true target is the conditional expectation $\mathbb{E}[Y|X = \mathbf{x}]$.[2]

Exercise 20: We can similarly compute the optimal predictor for the absolute error cost, in Equation (7.3). Show that the optimal predictor for the absolute error is the conditional median, Median$[Y|X = \mathbf{x}]$. □

7.3 Reducible and Irreducible Error

Having found the optimal regression model, we can now write the expected cost in the cases of both optimal and suboptimal models $f(\mathbf{x})$. That is, we are interested in expressing $\mathbb{E}[C]$ when

1. $f(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$
2. $f(\mathbf{x}) \neq \mathbb{E}[Y|\mathbf{x}]$.

When $f(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$, the expected cost can be simply expressed as

$$\begin{aligned}
\mathbb{E}[C] &= \int_{\mathcal{X}} p(\mathbf{x}) \int_{\mathcal{Y}} (\mathbb{E}[Y|\mathbf{x}] - y)^2 p(y|\mathbf{x})dyd\mathbf{x} \\
&= \int_{\mathcal{X}} p(\mathbf{x})\text{Var}\left[Y|X = \mathbf{x}\right] d\mathbf{x} \qquad (7.4)
\end{aligned}$$

Recall that Var $[Y|X = \mathbf{x}]$ is the variance of Y, for the given $\mathbf{x}$. The expected cost, therefore, reflects the cost incurred from noise or variability in the targets. This is the best scenario in regression for a squared error cost; we cannot achieve a lower expected cost.

[2]It may appear that in the above equations, setting $f(\mathbf{x}) = y$ would always lead to $\mathbb{E}[C] = 0$. Unfortunately, this would be an invalid operation because for a single input $\mathbf{x}$ there may be multiple possible outputs y and they can certainly appear in the same data set. To be a well-defined function, $f(\mathbf{x})$ must always have the same output for the same input. $\mathbb{E}[C] = 0$ can only be achieved if $p(y|\mathbf{x})$ is a delta function for every $\mathbf{x}$.

The next situation is when $f(\mathbf{x}) \neq \mathbb{E}[Y|\mathbf{x}]$. Here, we will proceed by decomposing the squared error as

$$\begin{aligned}(f(\mathbf{x}) - y)^2 &= (f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}] + \mathbb{E}[Y|\mathbf{x}] - y)^2 \\ &= (f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}])^2 + 2\underbrace{(f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}])(\mathbb{E}[Y|\mathbf{x}] - y)}_{g(\mathbf{x},y)} + (\mathbb{E}[Y|\mathbf{x}] - y)^2\end{aligned}$$

Notice that the expected value of $g(\mathbf{x}, Y)$ for each $\mathbf{x}$ is zero because

$$\begin{aligned}\mathbb{E}[g(\mathbf{x}, Y)] &= \mathbb{E}\Big[(f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}])(\mathbb{E}[Y|\mathbf{x}] - Y)|\mathbf{x}\Big] \\ &= (f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}])\mathbb{E}\Big[(\mathbb{E}[Y|\mathbf{x}] - Y)|\mathbf{x}\Big] \\ &= (f(\mathbf{x}) - \mathbb{E}[Y|\mathbf{x}])\,(\mathbb{E}[Y|\mathbf{x}] - \mathbb{E}[Y|\mathbf{x}]) \\ &= 0.\end{aligned}$$

Therefore, we can conclude that $\mathbb{E}[g(\boldsymbol{X}, Y)] = 0$, when taking expectations over $\boldsymbol{X}$. We can now express the expected cost as

$$\begin{aligned}\mathbb{E}[C] &= \mathbb{E}[(f(\boldsymbol{X}) - Y)^2] \\ &= \underbrace{\mathbb{E}[(f(\boldsymbol{X}) - \mathbb{E}[Y|\boldsymbol{X}])^2]}_{\text{reducible error}} + \underbrace{\mathbb{E}[(\mathbb{E}[Y|\boldsymbol{X}] - Y)^2]}_{\text{irreducible error}}.\end{aligned}$$

The first term reflects how far the trained model $f(\mathbf{x})$ is from the optimal model $\mathbb{E}[Y|\mathbf{x}]$. The second term reflects the inherent variability in Y given $\mathbf{x}$, as written in Equation (7.4). These terms are also often called the *reducible* and *irreducible* errors. If we extend the class of functions f to predict $\mathbb{E}[Y|\mathbf{x}]$, we can reduce the first expected error. However, the second error is inherent or irreducible in the sense that no matter how much we improve the function, we cannot reduce this term. This relates to the problem of partial observability, where there is always some stochasticity due to a lack of information. This irreducible distance could potentially be further reduced by providing more feature information (i.e., extending the information in $\mathbf{x}$). However, for a given dataset, with the given features, this error is irreducible.

To sum up, we argued here that optimal classification and regression models critically depend on knowing or accurately learning the posterior distribution $p(y|\mathbf{x})$. This task can be solved in different ways, but a straightforward approach is to assume a functional form for $p(y|\mathbf{x})$, say $p(y|\mathbf{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of weights or parameters that are to be learned from the data.

Remark: You may be wondering how these two errors relate to the notions of bias and variance. Bias and variance examines how much the learned estimator—which here is f—changes with different datasets. The above analysis was about decomposing errors for one given function f. We will see in Section 10.3 that the irreducible error, in expectation across all datasets of size n—and so across all the possible learned functions we could see across these datasets—corresponds to the squared bias plus the variance. There, we will reason that the expected mean-squared error for regression, where expectation is across different datasets we could have seen, decomposes into bias and variance (expected irreducible error) and reducible error.

Chapter 8

Linear Regression and Polynomial Regression

Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ the objective is to learn the relationship between features and the target. We start by hypothesizing the functional form of this relationship. For example, the function f might be a linear function

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2$$

where we learn $\mathbf{w} = (w_0, w_1, w_2)$. Alternatively, we might hypothesize that f is a nonlinear function, such as $f(x) = \alpha + \beta x_1 x_2$, where α and β need to be learned.

In this chapter, we focus on estimating linear functions. The function is modeled as a *linear combination* of features and parameters, i.e.

$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots w_d x_d = \sum_{j=0}^{d} w_j x_j = \mathbf{x}^\top \mathbf{w}$$

where we extended $\mathbf{x}$ to $(x_0 = 1, x_1, x_2, \ldots, x_d)$. This choice is for simplicity of notation: it allows us to write $f(\mathbf{x})$ as a dot product, and avoid having to specially account for the intercept term w_0. Finding the best parameters $\mathbf{w} \in \mathbb{R}^{d+1}$ is referred to as the *linear regression problem*. We begin by formalizing this as a maximum likelihood problem.

8.1 Maximum Likelihood Formulation

We consider a statistical formulation of linear regression. We assume the datapoints $\mathbf{x}_i$ are generated according to some (unknown) distribution $p(\mathbf{x})$. We assume that the target variable Y has an underlying linear relationship with input $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$, plus a noise term ε that follows a zero-mean Gaussian distribution, i.e. $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. That is, for a given input $\mathbf{x}$, the target y is a realization of a random variable Y defined as

$$Y = \sum_{j=0}^{d} \omega_j x_j + \varepsilon,$$

where $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_d)$ are the true underlying parameters, and $x_0 = 1$ is the intercept term. The assumption of normality for the error term is reasonable, although the independence between ε and $\boldsymbol{X}$ may not hold in practice.[1] We can see that Y given $\mathbf{x}$ also follows a Gaussian distribution, i.e. its conditional density is $p(y|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}^\top \mathbf{x}, \sigma^2)$.

[1] The justification for this comes from the central limit theorem. Y is obtained by summing up multiple random variables. The central limit theorem states that the normalized sum of independent random variables (need not be Gaussian) becomes more and more Gaussian with more variables in the sum. For a reasonable number of features, Y is approximately Gaussian, so it is reasonable to assume a Gaussian noise component.

In linear regression, we seek to approximate the target as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, where weights $\mathbf{w}$ are to be determined. We first write the conditional likelihood function for a single pair $(\mathbf{x}, y)$ as

$$p(y|\mathbf{x}, \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

where we use the notation $\exp(a) = e^a$, to make the exponent easier to read. Observe that the only change from the conditional density function of Y is that coefficients $\mathbf{w}$ are used instead of $\boldsymbol{\omega}$. The MLE problem, where we assume the space of possible values for the weights is $\mathcal{F} \subset \mathbb{R}^{d+1}$ is

$$\begin{aligned}
\mathbf{w}_{\text{MLE}} &= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} -\sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i, \mathbf{w}) \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} -\sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2\sigma^2}\right) \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} -\sum_{i=1}^{n} \left[-\ln\sqrt{2\pi\sigma^2} - \frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2\sigma^2}\right] \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} \sum_{i=1}^{n} \ln\sqrt{2\pi\sigma^2} + \sum_{i=1}^{n} \frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2\sigma^2} \quad \triangleright \text{ first term is constant w.r.t. } \mathbf{w} \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} \sum_{i=1}^{n} \frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2\sigma^2} \quad \triangleright \text{ dropping the first term does not change the solution} \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \mathbf{w})^2 \quad \triangleright \text{ scaling by } 2\sigma^2 \text{ does not change the solution} \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathcal{F}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \mathbf{w})^2 \qquad (8.1)
\end{aligned}$$

The resulting objective is intuitive. Our prediction $\hat{y}_i \stackrel{\text{def}}{=} \mathbf{x}_i^\top \mathbf{w}$. The MLE formulation states that we should find the weights that minimize the squared differences between our predictions $\hat{y}_i$ and the given y_i. A simple example illustrating the linear regression problem is shown in Figure 8.1. In the next sections, we will discuss how to solve this optimization and the properties of the solution.

Note that here it seems obvious that we could have just started with the squared error objective. This is in fact how ordinary least squares (OLS) was originally introduced. However, there are two reasons that we use the MLE approach. First, the statistical framework provides insights into the assumptions behind OLS regression. In particular, the assumptions include that the data $\mathcal{D}$ was drawn i.i.d.; there is an underlying linear relationship between features and the target; that the noise (error term) is zero-mean Gaussian and independent of the features; and that there is an absence of noise in the collection of features. Second, for other distributions $p(y|\mathbf{x})$—such as the Bernoulli when we do classification—guessing a good objective is much less obvious. For those situations, we will often turn to an MLE formulation to help us define a reasonable objective for our parameters. When possible, it is better to take a more unified approach.

Example 23: Consider again data set $\mathcal{D} = \{(1, 1.2), (2, 2.3), (3, 2.3), (4, 3.3)\}$ from Figure 8.1. We want to find the maximum likelihood coefficients—the least-squares fit—for $f(x) =$

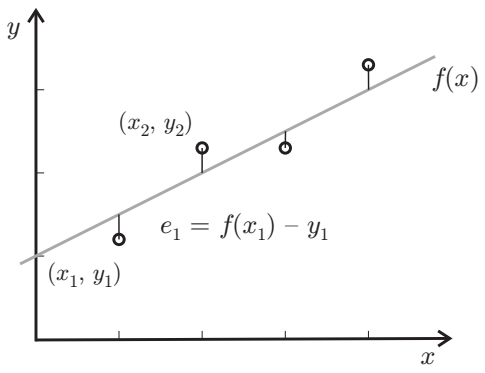


Figure 8.1: A linear regression solution on data set $\mathcal{D} = \{(1, 1.2), (2, 2.3), (3, 2.3), (4, 3.3)\}$. *The task of the optimization process is to find the best linear function* $f(x) = w_0 + w_1 x$ *so that the sum of squared errors* $e_1^2 + e_2^2 + e_3^2 + e_4^2$ *is minimized.*

$w_0 + w_1 x$. The problem corresponds to finding the solution with the following variables

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} 1.2 \\ 2.3 \\ 2.3 \\ 3.3 \end{bmatrix},$$

where a column of ones was added to $\mathbf{X}$ to allow for a non-zero intercept w_0. We can substitute $\mathbf{X}$ and $\mathbf{y}$ into Equation (8.3) below, to get the solution $\mathbf{w} = (0.7, 0.63)$. At this point, the gradient w.r.t. $\mathbf{w}$ of the sum of squared errors is zero and the sum of squared errors is 0.223. □

8.2 Linear Regression Solution

To minimize the sum of squared errors, let's define

$$c_i(\mathbf{w}) = \tfrac{1}{2} \left(f(\mathbf{x}_i) - y_i\right)^2 = \tfrac{1}{2} \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2, \tag{8.2}$$

with the full objective as

$$c(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} c_i(\mathbf{w})$$

We added a normalization by the number of samples, so that we have an average squared error rather than a cumulative error. This optimization is equivalent, since normalization by a fixed constant does not change the solution: $\operatorname{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} c_i(\mathbf{w}) = \operatorname{argmin}_{\mathbf{w}} \sum_{i=1}^{n} c_i(\mathbf{w})$. Numerically, though, the average error is more sensible to optimize, since it will not grow with more data. The cumulative error, on the other hand, can get very big if n is big. So, to implement a solution, we will use Equation (8.2) instead of Equation (8.1). Similarly, we used $\frac{1}{2}$ in front of Equation (8.2) without changing the solution; this will be convenient later to cancel the 2 that comes from the gradient.

Notice that to compute the gradient of the objective, it decomposes into a sum of gradients for each sample

$$\nabla c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla c_i(\mathbf{w})$$

and so we simply need to determine the gradient of the error for each sample $(\mathbf{x}_i, y_i)$. We can use the chain rule, by introducing variable $u = \mathbf{x}_i^\top \mathbf{w} - y_i$, to get

$$\begin{aligned}
\frac{\partial c_i(\mathbf{w})}{\partial w_j} &= \frac{\partial \frac{1}{2}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2}{\partial w_j} \\
&= \frac{\partial \frac{1}{2}u^2}{\partial u}\frac{\partial u}{\partial w_j} && \triangleright\ u \stackrel{\text{def}}{=} \mathbf{x}_i^\top \mathbf{w} - y_i \\
&= u\frac{\partial(\sum_{m=0}^{d} x_{im}w_m - y_i)}{\partial w_j} \\
&= u\sum_{m=0}^{d}\frac{\partial x_{im}w_m}{\partial w_j} = ux_{ij} && \triangleright\ \frac{\partial x_{im}w_m}{\partial w_j} = 0 \text{ for } m \neq j \\
&= (\mathbf{x}_i^\top \mathbf{w} - y_i)x_{ij}
\end{aligned}$$

This derivation is for any $0 \leq j \leq d$.

Our goal is to find $\mathbf{w}$ such that $\frac{\partial c(\mathbf{w})}{\partial w_j} = 0$ for all $0 \leq j \leq d$. We obtain a system of equations, with $d+1$ variables and $d+1$ equations

$$\begin{aligned}
\tfrac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)x_{i0} &= 0 \\
\tfrac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)x_{i1} &= 0 \\
&\vdots \\
\tfrac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)x_{id} &= 0
\end{aligned}$$

which can be equivalently written in vector notation as $\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)\mathbf{x}_i = \mathbf{0}$, where $\mathbf{0}$ is the $d+1$ vector of all zeros. We can turn to linear algebra to obtain a solution. We can write down the matrix and vector that correspond to this linear system of equations

$$\begin{aligned}
\mathbf{A} &\stackrel{\text{def}}{=} \tfrac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top \in \mathbb{R}^{(d+1)\times(d+1)} \\
\mathbf{b} &\stackrel{\text{def}}{=} \tfrac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i y_i \in \mathbb{R}^{d+1}
\end{aligned} \tag{8.3}$$

Then our goal is to find $\mathbf{w}$ such that $\mathbf{A}\mathbf{w} = \mathbf{b}$. If $\mathbf{A}$ is invertible, then $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$.

In practice, though, it is more common to solve for the weights $\mathbf{w}$ using gradient descent. For gradient descent, we would initialize the weights $\mathbf{w}$ at some random initialization and iteratively update $\mathbf{w}$ until we reach a point where the gradient is approximately zero. This is shown in the pseudocode, in Algorithm 4.

Algorithm 4: Gradient Descent for a given objective c

Fix iteration parameters: tolerance $= 10^{-4}$ and max iterations $= 10^5$
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
err $\leftarrow \infty$
while $|c(\mathbf{w}) - \text{err}| >$ tolerance and have not reached max iterations **do**
  err $\leftarrow c(\mathbf{w})$ ▷ for linear regression, $c(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$
  $\mathbf{g} \leftarrow \nabla c(\mathbf{w})$ ▷ for linear regression, $\nabla c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)\mathbf{x}_i$
  // The step-size η could be chosen by line-search, as in Algorithm 1
  $\eta \leftarrow$ line search$(\mathbf{w}, c, \mathbf{g})$
  $\mathbf{w} \leftarrow \mathbf{w} - \eta\mathbf{g}$
return $\mathbf{w}$

Using gradient descent can be more efficient than solving for $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$. Each gradient descent update costs $O(nd)$, whereas the solution to the linear system costs $O(d^3 + d^2n)$. It costs d^2n to construct $\mathbf{A}$ and matrix inversion[2] costs approximately $O(d^3)$. If we use k iterations for gradient descent, then gradient descent costs $O(ndk)$ in total. For big d and fewer samples, gradient descent is almost definitely more efficient. Further, we can get to an approximate solution with gradient descent efficiently, with a relatively small number of iterations.

The cost of gradient descent, though, is still quite high because we have to iterate over the entire dataset to compute the gradient. It is not uncommon to have very large datasets. Rather, what is really used in practice is the much more efficient *stochastic gradient descent*, which we discussed in Chapter 6. In fact, the mini-batch SGD algorithm given in Algorithm 3 has the specific update for linear regression.

8.3 Polynomial Regression: Using Linear Regression to Learn Non-linear Predictors

At first, it might seem that the applicability of linear regression to real-life problems is greatly limited. After all, it is not clear whether it is realistic (most of the time) to assume that the target variable is a linear combination of features. Fortunately, the applicability of linear regression is broader because we can use it to obtain non-linear functions. The main idea is to apply a non-linear transformation to the observation vector $\mathbf{x}$ prior to the fitting step. A linear function in this new feature space provides a nonlinear function in the original observation space. In this section, we will discuss one such nonlinear transformation: polynomials. Many other nonlinear transformations are possible—see for example radial basis functions, wavelets, and Fourier basis to name a few. The idea, though, is the same, and so we will use polynomials as one representative example.

Let's start by considering one-dimensional data, i.e., $d = 1$. In OLS regression, we would learn the function

$$f(x) = w_0 + w_1 x,$$

[2]We can actually use other linear system solvers to find $\mathbf{w}$ such that $\mathbf{A}\mathbf{w} = \mathbf{b}$; the best strategy is not necessarily to compute the matrix inverse. But, the costs are still typically close to $O(d^3)$, so for simplicity of analysis we use this computational complexity.

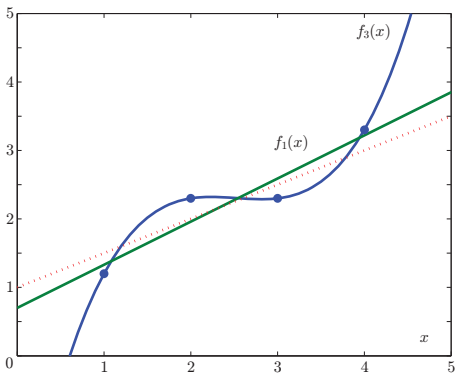


Figure 8.2: Example of a linear vs. polynomial fit on a data set shown in Figure 8.1. The linear fit, $f_1(x)$, is shown as a solid green line, whereas the cubic polynomial fit, $f_3(x)$, is shown as a solid blue line. The dotted red line indicates the target linear concept.

where x is the data point and $\mathbf{w} = (w_0, w_1)$ is the weight vector. To achieve a polynomial fit of degree p, we will modify the previous expression into

$$f(x) = \sum_{j=0}^{p} w_j x^j,$$

where p is the degree of the polynomial. We will rewrite this expression using a set of basis functions as

$$f(x) = \sum_{j=0}^{p} w_j \phi_j(x) = \mathbf{w}^\top \boldsymbol{\phi},$$

where $\phi_j(x) = x^j$ and $\boldsymbol{\phi} = (\phi_0(x), \phi_1(x), \ldots, \phi_p(x))$. We simply apply this transformation to every data point x_i to get a new dataset $\{(\phi(x_i), y_i)\}$. Then we use linear regression on this dataset, to get the weights $\mathbf{w}$ and the nonlinear predictor $f(x) = \sum_{j=0}^{p} w_j \phi_j(x)$, which is a polynomial (nonlinear) function in the original observation space.

Example 24: In Figure 8.1 we presented an example of a data set with four data points. What we did not mention was that, given a set $\{x_1, x_2, x_3, x_4\}$, the targets were generated by using function $1 + \frac{x}{2}$ and then adding a measurement error $\boldsymbol{\epsilon} = (-0.3, 0.3, -0.2, 0.3)$. It turned out that the optimal coefficients $\mathbf{w}_{\text{MLE}} = (0.7, 0.63)$ were close to the true coefficients $\boldsymbol{\omega} = (1, 0.5)$, even though the error terms were relatively significant. We will now attempt to estimate the coefficients of a polynomial fit with degrees $p = 2$ and $p = 3$.

First let us consider how we get these new functions. For $p = 2$, we get $f_2(x) = w_0 + w_1 x + w_2 x^2$, or in other words we have

$$\boldsymbol{\phi}_2(x) = \begin{bmatrix} \phi_0(x) = 1.0 \\ \phi_1(x) = x \\ \phi_2(x) = x^2 \end{bmatrix} \quad \text{e.g., } \boldsymbol{\phi}_2(1) = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}, \ \boldsymbol{\phi}_2(2) = \begin{bmatrix} 1.0 \\ 2.0 \\ 4.0 \end{bmatrix}, \ \boldsymbol{\phi}_2(3) = \begin{bmatrix} 1.0 \\ 3.0 \\ 9.0 \end{bmatrix}$$

with $f_2(x) = \boldsymbol{\phi}_2(x)^\top \mathbf{w}_2$ for $\mathbf{w}_2 \stackrel{\text{def}}{=} [w_0, w_1, w_2]^\top$. For $p = 3$, we get $f_3(x) = \tilde{w}_0 + \tilde{w}_1 x +$

$\tilde{w}_2 x^2 + \tilde{w}_3 x^3$, or in other words we have

$$\boldsymbol{\phi}_3(x) = \begin{bmatrix} \phi_0(x) = 1.0 \\ \phi_1(x) = x \\ \phi_2(x) = x^2 \\ \phi_3(x) = x^3 \end{bmatrix} \quad \text{e.g., } \boldsymbol{\phi}_3(1) = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}, \ \boldsymbol{\phi}_2(2) = \begin{bmatrix} 1.0 \\ 2.0 \\ 4.0 \\ 8.0 \end{bmatrix}, \ \boldsymbol{\phi}_2(3) = \begin{bmatrix} 1.0 \\ 3.0 \\ 9.0 \\ 27.0 \end{bmatrix}$$

with $f_3(x) = \boldsymbol{\phi}_3(x)^\top \mathbf{w}_3$ for $\mathbf{w}_3 \stackrel{\text{def}}{=} [\tilde{w}_0, \tilde{w}_1, \tilde{w}_2, \tilde{w}_3]^\top$. We can use linear regression on these new features, $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$ to get the best polynomial fits for $\mathbf{w}_2$ and $\mathbf{w}_3$ respectively.

Using a polynomial fit with degrees $p = 2$ and $p = 3$ results in $\mathbf{w}_2 = (0.575, 0.755, -0.025)$ and $\mathbf{w}_3 = (-3.1, 6.6, -2.65, 0.35)$, respectively. The average squared error on the dataset is $c(\mathbf{w}_2) = 0.055$ and $c(\mathbf{w}_3) \approx 0$. Thus, the best fit is achieved with the cubic polynomial.

Note that this polynomial is strictly more expressive than either the linear function and the degree two polynomial, since it can always choose to set the weights to zero and ignore the added features. It is no surprise, then, that it can achieve a lower squared error than either the linear function or the quadratic function. As we discuss in the next chapter, however, this cubic function is actually not a good predictor. It can achieve a better fit, but does not generalize well to new data nor does it discover the true linear function that generated the data. □

The same idea extends to multivariate observation vectors $\mathbf{x}$. Each transformation $\phi_j : \mathcal{X} \to \mathbb{R}$ produces one of the terms $\phi_j(\mathbf{x})$ for the polynomial for the given input $\mathbf{x}$. For example, for $\mathbf{x} = (x_1, x_2)$, we could define polynomial basis

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_0(\mathbf{x}) = 1.0 \\ \phi_1(\mathbf{x}) = x_1 \\ \phi_2(\mathbf{x}) = x_2 \\ \phi_3(\mathbf{x}) = x_1 x_2 \\ \phi_4(\mathbf{x}) = x_1^2 \\ \phi_5(\mathbf{x}) = x_2^2 \end{bmatrix}$$

This transformation allows us to learn a degree-2 polynomial, now with 6 variables $w_0, \ldots, w_5$ to learn rather than the 3 we would learn for $(x_0 = 1, x_1, x_2)$. The resulting function is a polynomial with coefficients w_j:

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 = \sum_{j=0}^{6} w_j \phi_j(\mathbf{x}).$$

We can similarly obtain a cubic functions, adding the required terms for degree-3 polynomial, including $x_1^2 x_2$, $x_1 x_2^2$, x_1^3 and so on. The number of variables for this cubic, if we include all the possible terms, is 10.

In general, for a p-degree polynomial on d inputs, the number of terms corresponds to number of combinations of choosing p elements from a set of $d+1$ elements[3] if repetition is allowed: $\binom{(d+1)+p-1}{p} = \binom{d+p}{p}$. So, for two variables, the quadratic has $\binom{2+2}{2} = \binom{4}{2} = 6$; the cubic has $\binom{2+3}{3} = \binom{5}{3} = 10$; and a quartic would have $\binom{2+4}{4} = \binom{6}{4} = 15$. For three variables, the quadratic has $\binom{3+2}{2} = \binom{5}{2} = 10$; the cubic has $\binom{3+3}{3} = \binom{6}{3} = 20$; and a quartic would

[3] It is $d+1$ because in the polynomial input we consider $x_0 = 1.0$ to be a valid term to select.

have $\binom{3+4}{4} = \binom{7}{4} = 35$. If we let $k = \binom{(d+1)+p-1}{p}$, then the resulting polynomial can be generically written

$$f(\mathbf{x}) = \sum_{j=0}^{k} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}),$$

and the new number of features for linear regression is k.

Remark: Recall that in linear regression, we made the assumption that Y given an input $\mathbf{x}$ is Gaussian distributed: $Y \sim \mathcal{N}(\mu = \mathbf{x}^\top \mathbf{w}, \sigma^2)$. The distribution of the observation vector itself was not relevant. Under this nonlinear transformation, we can similarly notice we are making a Gaussian assumption. However, now the mean for the Gaussian is a more complex, nonlinear function of $\mathbf{x}$: $Y \sim \mathcal{N}(\mu = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}, \sigma^2)$.

Exercise 21: Write the SGD update for polynomial regression. □

Chapter 9

Generalization Error and Evaluation of Models

The majority of this book has focused on algorithm derivation and obtaining models, but we have yet to address how to evaluate these models. The maximum likelihood formalism for deriving learning algorithms provides some consistency results, where in the limit of samples we can discuss the convergence point of an estimator. In practice, however, we would like to evaluate the models and the algorithms based on a finite sample. Imagine a setting where you learn two models, say using linear regression and polynomial regression. Which of these two models is "better"? What does it even mean to say better? Are we trying to compare algorithms or models obtained from a specific instance of an algorithm? How can we be confident that the measured performance accurately reflects the performance we expect to see on new data? These questions are largely separate from our previous questions of effectively optimizing a specified objective, and rather starts to ask questions about the properties of that objective and about empirical properties of learned models.

In this chapter, we provide empirical tools to better evaluate the properties of learning algorithms and models. We will start by discussing the idea of generalization error: the expected cost of a model across all possible datapoints. This reflects how well that model generalizes to data it did not get trained on, which reflects our ultimate goal of deploying predictors to make predictions on new data. We will discuss the nuances in trying to accurately estimate this generalization error. Towards this goal, we will discuss how to split data and how to use statistical significance tests to provide some level of confidence that one algorithm or model is better than another, under some specific criteria. We will rarely be able to make strong conclusions based on experiments, but we can build up some evidence on the algorithm or model's properties.

These tools are arguably the most critical aspects of properly using machine learning algorithms in practice. One can learn a complex model, but without any understanding of how it is expected to perform in practice on new data, it is not viable to actually use these models. Whether an algorithm is used for scientific purposes or deployed in real systems, having an understanding of its properties both theoretically and empirically is key to obtain expected outcomes. This chapter only scratches the surface of the complex topic of proper evaluation of learning algorithms and models. For a nice overview of evaluation for machine learning algorithms, see [9].

9.1 Generalization Error, Overfitting and Underfitting

Our goal is to minimize generalization error: the expected cost. Recall that for a given loss, or cost $\text{cost}(\hat{y}, y)$, the expected cost is

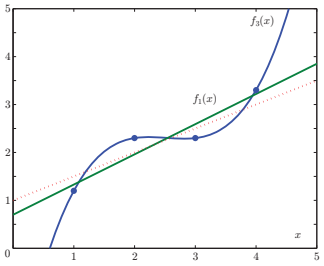
$$\mathbb{E}[C] = \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\text{cost}(f(\mathbf{x}), y)d\mathbf{x}dy.$$

We already saw this goal when talking about optimal predictors. But, we cannot directly minimize the generalization error; instead, we have minimized an empirical error. For example, for regression, we minimized a sample average squared error $\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2$ which is an unbiased and consistent estimate of the true expected cost. A natural question is, for this minimum $\mathbf{w}$ of the empirical cost on the given training dataset, how well does it do in terms of the expected cost?

One might hope that by minimizing the empirical cost, that we should do well in terms of the expected cost. After all, a sample average is a reasonable estimate of the expectation. Unfortunately, in some cases, minimizing the empirical cost can produce functions that generalize poorly. Consider the following extreme example, where f is a function that memorizes the data. For every observed $\mathbf{x}_i$, it returns precisely y_i. For any $\mathbf{x}_i$ that are not observed, it returns 0. This function will get zero empirical error—also called training error—but will not generalize at all and is likely to do very poorly in terms of the expected cost—the generalization error.

This effect is called *overfitting*. The term comes from the idea that the function has overly specialized—or overly fit itself—to the training set, at the expense of performing well for other data points. Overfitting typically occurs because the complexity of the model is increased considerably, whereas the size of the data set remains small relative to this model size. Even with modern datasets that are quite large, model complexity has increased at a commiserate pace, and so these very large models can overfit to the large datasets.

Example 25: Let's return to the polynomial regression example, shown in Figure 8.2 and which we redisplay here for convenience. Now, instead of simply asking how well the function can fit the data, we ask how well it performs on new data. To evaluate the learned models, we generate a testing dataset of 100 samples with observations $x \in \{0, 0.1, 0.2, \ldots, 10\}$ and noise-free target values generated using the true function $1 + \frac{x}{2}$.



Using a polynomial fit with degrees $p = 2$ and $p = 3$ results in $\mathbf{w}_2 = (0.575, 0.755, -0.025)$ and $\mathbf{w}_3 = (-3.1, 6.6, -2.65, 0.35)$, respectively. The average error on the training dataset equals $c(\mathbf{w}_2) = 0.05$ and $c(\mathbf{w}_3) \approx 0$. Thus, the best fit is achieved with the cubic polynomial. However, the average error on the test dataset reveals poor generalization ability of the cubic model. The average squared errors are $c_{\text{test}}(\mathbf{w}) = 0.269$, $c_{\text{test}}(\mathbf{w}_2) = 0.039$, and $c_{\text{test}}(\mathbf{w}_3) = 220.185$. It is clear the cubic model is overfitting, which is not surprising considering it has much higher model complexity than the linear models—and higher than is required to actually fit the data. □

Though we know overfitting occurs, it is not always obvious to see that it is occurring. One signature of overfitting is an increase in the magnitude of the coefficients. This manifests in the above example. While the absolute values of all coefficients in $\mathbf{w}$ and $\mathbf{w}_2$ were less than one, the values of the coefficients in $\mathbf{w}_3$ became significantly larger with alternating signs. (We will discuss *regularization* in Chapter 10 as an approach to prevent this effect.) This occurs because the cubic function has four unknowns (four parameters) and only four observations: it can fit this small dataset of four examples perfectly. In particular, it can fit the noise ϵ in the targets by adding and subtracting large numbers to get the precise y values. For sample $(x = 1, y = 1.2)$, we have $f(x) = -3.1 + 6.6 * 1 - 2.65 * 1^2 + 0.35 * 1^3 = 1.2$.

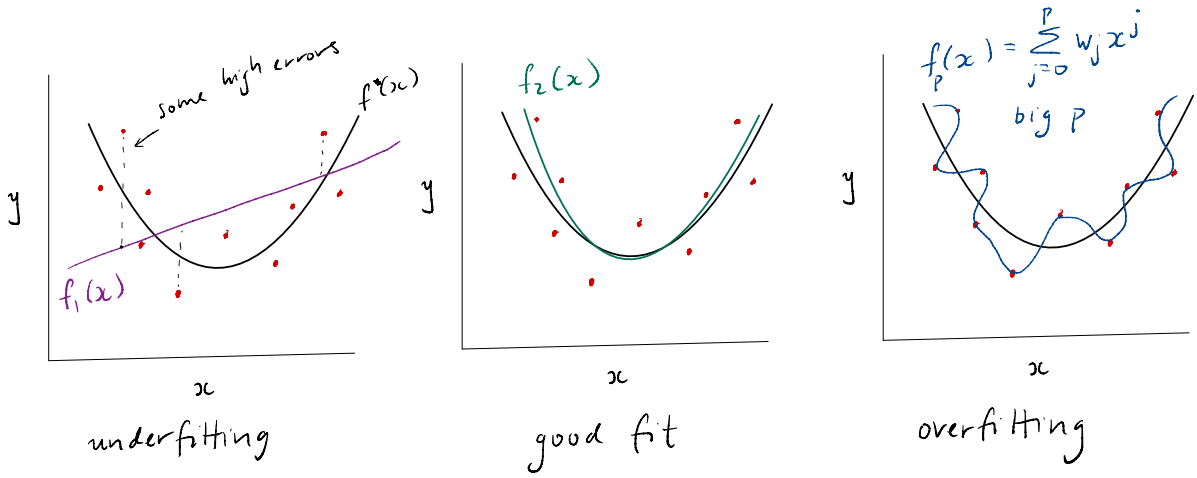


Figure 9.1: The true data is generated according to f^*, the quadratic in black, with observed datapoints in red. Each datapoint is generated using $y = f^*(x) + \epsilon$ for some zero-mean Gaussian noise ϵ. The linear function f_1 cannot fit these points that have a quadratic shape, and so has high error. The quadratic function f_2 fits the points well. When p is very big, the points can be fit exactly, but the function is clearly overfit to these given points.

For sample $(x = 2, y = 2.3)$, we have $f(x) = -3.1 + 6.6 * 2 - 2.65 * 2^2 + 0.35 * 2^3 = 2.3$. And so on. The function contorts to an odd solution, so that it can perfectly match the given y, which may actually match the noise in y rather than our actual target, $\mathbb{E}[Y|x]$.

Models may not only suffer from overfitting; they can also suffer from *underfitting*. The problem here is usually that the model class is insufficient to represent the true model, and so is not able to obtain a good fit. In this case, the training error can be quite high. Another way to see this is that the learned model cannot explain the data. It is reasonable to expect that with growing dataset sizes, for increasingly complex problems, our models will begin to suffer more from underfitting than overfitting. The true model—determined by complex interactions in the world—is likely not in our function class. We visualize underfitting and overfitting, for a true model that is quadratic, in Figure 9.1.

9.2 Estimating Generalization Error with Test Sets

We can diagnose overfitting by obtaining samples of generalization error. The most straightforward way to obtain a measure of the generalization error for a model is to use a test set. Before doing any training, a part of the data is set to the side—or held out—to only be used at the very end to gauge performance of our learned function. For example, if we have 10,000 samples, we could use $n = 8000$ for training and $m = 2000$ for testing. We train f on the first $n = 8000$ datapoints and then measure

$$\text{GE}(f) \approx \text{Test-Error}(f) = \frac{1}{m} \sum_{i=n+1}^{n+m} \text{cost}(f(\mathbf{x}_i), y_i)$$

For example, for regression, we used $\text{cost}(f(\mathbf{x}_i), y_i) = (f(\mathbf{x}_i) - y_i)^2$.

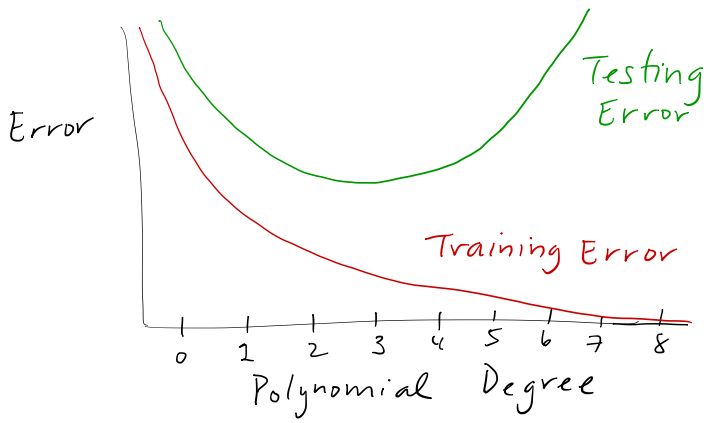


Figure 9.2: Hypothetical training and testing error, for polynomial regression with increasing polynomial degree. The training error decreases with increasing degree (increasing model complexity). The testing error improves, when using more complex models, but then starts to degrade with higher-order polynomials due to overfitting.

Once we have these m samples of error, we can try to make statistically sound conclusions about performance. We can use confidence intervals, with the sample average error on the test set, to gauge the level of certainty in this estimate of the generalization error. If our confidence intervals are narrow, then we can be relatively confident in our estimate. If the confidence intervals are quite wide, then we need to exercise caution using our estimate and it might be worthwhile gathering more data for testing before any deployment. Further, we can also use these m estimates of error to compare different models, say one using polynomial regression with $p = 2$ and another with $p = 3$. We discuss using statistical significance tests to make high-probability claims about differences between algorithms, in the next section.

One disadvantage of using a held-out test set to estimate GE is that we cannot use all the data for training. But, we always want more training data. The naive approach would be to simply train on all the data, and then use this training error as an estimate of the GE. Unfortunately, this would be terribly biased. Consider again a function that perfectly overfits the data: it would zero training error, but likely high GE. In fact, to give an unbiased sample of generalization error, the hold-out test set cannot be used in any way during training. There are, however, clever ways to try to split the data—called cross validation—to allow us to get reasonable estimates of the performance of the model trained on all the data. We will use the simpler test set approach here, but for more about these other strategies see the thorough and accessible explanation in [8, Chapter 5].

Now let us consider how to use this test set to diagnose overfitting. If there is a significant mismatch between training and testing error, this could indicate overfitting. We cannot simply compare the numbers, because we generally expect training error to be lower than testing error (and lower than generalization error). The weights were chosen to minimize that error, after all. A function could be quite good in terms of generalization error, and still have lower training error than testing error. Instead, it is typically easier to gauge overfitting by comparing different models. For example, you could increase model complexity, such as

trying all polynomials up to degree p, and then see at what degree the testing error stops decreasing and starts increasing, as in Figure 9.2.

Notice that the training error can only decrease with increasing p because, for $\mathcal{F}_p$ the set of degree p polynomials,

$$\min_{f \in \mathcal{F}_{p+1}} \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2 = \min \Big(\min_{f \in \mathcal{F}_p} \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2, \min_{f \in \mathcal{F}_{p+1} \setminus \mathcal{F}_p} \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2 \Big)$$
$$\leq \min_{f \in \mathcal{F}_p} \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$$

Another way to see this is by noting that it is strictly more flexible to pick polynomials of higher degrees, since we can always set the coefficients for higher-order terms to zero. For example, for scalar inputs $x \in \mathbb{R}$, we can always take a $p+1$-order polynomials $f_{p+1}(x) = \sum_{j=0}^{p+1} w_j x^j$ and recover a p-order polynomial by fixing $w_{p+1} = 0$. You can see f_p as a polynomial that is forced to use w_{p+1}, whereas f_{p+1} can more flexibly use any value it wants for w_{p+1}. This flexibility means it should be better able to minimize the loss.

This approach can also be used to diagnose underfitting. We can see that when we increase p from 1 to 2, both the training error and testing error decrease. We cannot say for sure that the ideal p is at 3, with underfitting occurring at $p < 3$ and overfitting at $p > 3$. The test set, after all, only gives an estimate of the generalization error. However, with a sufficiently large test set, such a curve is likely quite representative of the performance of these models.

Remark about terminology: A common **mistake** is to think that the generalization error is the *gap* between the training and test error. This is incorrect. The generalization error is the expected cost, namely the *expected* test error. A large gap between training and test error can be indicative of poor generalization error, since it can indicate overfitting. We might use this gap to diagnose overfitting. But the estimate of generalization error that we ultimately care about is the test error. For example, a random predictor might have very similar performance on training and test, but it is equally bad performance. This random predictor has high generalization error (high test error), even though the generalization gap is near zero.

9.3 Making Statistically Significant Claims

Now that we have a mechanism to obtain m (unbiased) samples of error, we can turn to obtaining statistically significant (high-probability) claims about the performance of models. Suppose we have m samples, and wish to compare learned functions f_1 and f_2. When measuring performance on these m samples, we find that f_1 seems to perform better on average than f_2. But can we say that it is actually better? In this section, we discuss this questions, that is how to claim that f_1 is better than f_2, with high-probability, or realize that we cannot make such claims.

9.3.1 Computing Confidence Intervals Tests

One of the easiest approaches for evaluating and comparing models is to use a tool we already know: confidence intervals. We can compute a sample average error for a learned

function f, using the m samples of error from the test set $\mathcal{D}_{\text{test}} = \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$

$$\bar{X} = \frac{1}{m}\sum_{i=1}^{m} c_i(f)$$

where $c_i(f)$ is the error for the ith sample. For example, we could have a learned linear function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ and have squared error $c_i(f) = (\boldsymbol{X}_i^\top \mathbf{w} - Y_i)^2$. We know that we have an unbiased sample average, and so $\mathbb{E}[\bar{X}] = \text{GE}(f)$. Then we can obtain a $1-\delta$ (say 95%) confidence interval $\text{GE}(f) \in [\bar{X} - \epsilon, \bar{X} + \epsilon]$, where ϵ is such that $\Pr\left(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \geq \epsilon\right) \leq \delta$. If we believed the errors were Gaussian distributed with known variance σ^2, we could use a 95% Gaussian confidence interval, with $\epsilon = 1.96\sigma/\sqrt{m}$.

We liked the Gaussian interval because it required fewer samples to get a tighter interval; but sadly it is not really usable in practice. You might even be able to plot your m errors and notice they look Gaussian distributed. But, you definitely do not know their true variance. The *Student's t-distribution* is precisely designed for this setting. This distribution allows the sample variance to be used instead of the true variance. A 95% confidence interval is given by $\epsilon = t_{\delta,m-1} S_m/\sqrt{m}$ with the unbiased (Bessel-corrected) sample variance $S_m^2 = \frac{1}{m}\sum_{i=1}^m (c_i(f) - \bar{X})^2$. The constant $t_{\delta,m-1}$ now also depends on the number of samples, contrasting this constant for the Gaussian (which was 1.96). For $m = 2$, we have $t_{0.05,1} = 12.71$; for $m = 11$, we have $t_{0.05,10} = 2.228$; and for $m = 101$, we have $t_{0.05,100} = 1.984$. In the limit, as $m \to \infty$, this constant approach 1.96, because the distribution becomes a Gaussian distribution.

This confidence interval is useful just for evaluating the performance of one model. But, we can also use them to compare two models. If two intervals do not overlap, then we can say that they are different with high probability: that they are statistically significantly different. Assume f_1 has error $\bar{X}_1$ and f_2 has error $\bar{X}_2$, with correspondingly intervals given by ϵ_1 and ϵ_2. Then if $\bar{X}_1 + \epsilon_1 < \bar{X}_2 - \epsilon_2$, then we can say that f_1 is statistically significantly better than f_2 with confidence level δ.

9.3.2 Parametric Tests

Using confidence intervals is one of the simplest, but also least powerful statistical significance tests. We can do better by considering tests designed to compare two means. For now, let's start with a simple case, where we compare two models using the binomial test. Imagine you do not care about precise errors, but rather just want to rank the algorithms by saying which did better or worse. We can carry out such a comparison using a counting test: for each sample i, we award a win to f_1 if it has lower error and vice versa. In the case of exactly the same performance, we can provide a win/loss randomly.

	1	2	3	4		$m-1$	m
f_1	1	0	1	1	$\cdots$	0	1
f_2	0	1	0	0		1	0

Table 9.1: A counting test where models f_1 and f_2 are compared on a set of m independent samples. A model with better performance on a particular sample collects a win (1), whereas the other algorithm collects a loss (0).

Our goal is to provide statistical evidence that say model f_1 is better than model f_2. Suppose f_1 has k wins out of m and f_2 has $m-k$ wins, as shown in Table 9.1. Assume $k > m-k$, as otherwise we would be asking if f_2 is statistically significantly better than f_1. We would like to evaluate the null hypothesis H_0 that f_1 and f_2 have the same performance by providing an alternative hypothesis H_1 that f_1 is better than f_2. In short,

$$H_0\text{: quality}(f_1) = \text{quality}(f_2)$$
$$H_1\text{: quality}(f_1) > \text{quality}(f_2)$$

If the null hypothesis is true, the win/loss on each data set will be equally likely and determined by minor variation. Therefore, the probability of a win on any data set will be roughly equal to $\beta = 1/2$. Now, we can express the probability that f_1 would have collected k wins or more under the null hypothesis using the binomial distribution

$$p = \Pr(f_1 \text{ gets at least } k \text{ wins}) = \sum_{i=k}^{m} \binom{m}{i} \beta^i (1-\beta)^{m-i}$$

This value is the probability of k wins, plus the probability of $k+1$ wins, up to the probability of m wins, under the null hypothesis. It reflects how likely it is that f_1 would have been able to get so many wins, i.e., get a least k wins.

This probability p is referred to as the p-value. A typical approach in these cases is to establish a significance value, say, $\alpha = 0.05$ and reject the null hypothesis if $p \leq \alpha$. For sufficiently low p-values, we may conclude that there is sufficient evidence that f_1 is better than f_2. The p-value represents the likelihood of observing these outcomes—observing the evidence—if the null hypothesis is true. If the p-value is very small, this says that the probability of that evidence is very small, and so it suggests you were wrong to think the null hypothesis accurately describes the world. Instead, it is more reasonable to conclude your model of the world—the null hypothesis—is wrong. If the p-value is greater than α we say that there is insufficient evidence for rejecting[1] H_0.

The choice of the significance threshold α is somewhat arbitrary. A value of 5% is typical, but lower values indicate that the particular situation of k wins out of m was so unlikely, that we can consider the evidence for rejecting H_0 very strong. Being able to reject the null hypothesis provides some confidence that the result did not occur by chance.

More generally, we can consider other statistical significance tests based on the distributions of the performance measures. In the above example, a binomial distribution was appropriate. If instead we considered the actual errors on the datasets, then we have pairs of real values. In this case, a common choice is the *paired t-test*. This test can be used if both errors appear to be distributed normally and if they have similar variance. The paired t-test takes in the sampled differences between the algorithms (line 3 in Table 9.2), $d_1, \ldots, d_m$. Because again our null hypothesis is that the algorithms perform equally, under the null hypothesis the mean of these differences is 0. If the differences are normally distributed, then for the sample average $\bar{d} = \frac{1}{m}\sum_{i=1}^{m} d_i$ and sample standard deviation $S_d = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(d_i - \bar{d})^2}$, the random variable $t = \frac{\bar{d}-0}{S_d/\sqrt{m}}$ is distributed according to the

[1] Note that this does not mean that we accept the null hypothesis. Rather, we assumed it was true and evidence did not contradict that fact. But, absence of evidence is not a proof: just because it was not disproved, does not mean that it was proved.

Student's t-distribution. The Student's t-distribution is approximately like a normal distribution, with a degrees-of-freedom parameter $m-1$ that makes the distribution look more like a normal distribution as m becomes larger.

We can now ask about the probability of this random variable T, relative to the computed statistic. If we only care about knowing if algorithm 1 is better than algorithm 2, we conduct a one-tailed test. If the probability that T is larger than t, i.e., $p = Pr(T > t)$, is small, then we obtain some evidence that algorithm 1 is better than algorithm 2. To test if algorithm 1 is better than algorithm 2, we can swap the order of the difference. Or, in this setting, t would be negative, so we check if $p = Pr(T > -t)$ is small; then we obtain some evidence that algorithm 2 is better than algorithm 1. These are both one-tailed tests, reflecting the probabilities at one end of the tails of the distribution. A two-tailed test instead asks if the two algorithms are different; in this case, one would use $p = Pr(T > |t|)$. Note that this two-tailed test will always have a bigger p-value than the one-sided tests.

	1	2	3	4		$m-1$	m
f_1	0.11	0.08	0.15	0.12	$\cdots$	0.07	0.09
f_2	0.10	0.09	0.11	0.12	$\cdots$	0.10	0.09
performance difference d	0.01	-0.01	0.04	0.0	$\cdots$	-0.03	0.0

Table 9.2: A table of errors for two learning algorithms a_1 and a_2 are compared on a set of m independent data sets. The last row contains the differences, i.e., d = performance(f_1) - performance(f_1). These differences are used for the paired t-test.

9.3.3 How to Choose the Statistical Significance Test

We gave two examples of tests: the binomial test and the paired t-test. These tests make parametric assumptions. The binomial test requires that the compared values are 0, 1 values. The paired t-test assumes the difference in errors follows a Student's t-distribution, which is satisfied if the paired samples are normally distributed with equal variance. However, these conditions are not always satisfied, in which case other tests are more suitable. Further, there are some tests that do not make distributional assumptions, and rather are non-parametric. For a summary on selecting tests, see [9, Section 6.3]

The choice of the test comes down to satisfying assumptions, and the power of the test. The power of the test is the ability for the test to reject the null hypothesis, if it should be rejected. The approach using non-overlapping confidence intervals is a low-power test, because it does not take into account the paired errors for the two models. When a test fails to reject the null hypothesis, when it should have been rejected, this is called a Type II error (a false negative outcome). If the test is a low-power test, then it is more likely to commit a Type II error.

On the other hand, if a test is used when assumptions are violated, then we might falsely conclude that we can reject the null hypothesis, when in fact we should not have. This is called a Type I error (a false positive outcome). For example, if we make a relatively strong parametric assumption that the errors are Gaussian, then we have more power to reject the null hypothesis but might commit a Type I error if the errors are in fact not normally distributed.

These choices for statistical tests are similar to the choices we make when learning models: strong assumptions can enable faster learning, but are more biased and can lead to poorer predictions, whereas very general models can produce accurate predictions, but might need a lot of data. This choice is even more difficult for statistical significance tests, where the amount of available data is often highly limited—running experiments is expensive.

Chapter 10

Regularization and Constraining the Hypothesis Space

In this chapter, we discuss how regularization can be used to mitigate issues of overfitting. In particular, we discuss both regularizing the weights, as well as restricting the function class. We then discuss a foundational concept in machine learning: the bias-variance trade-off.

10.1 Regularization as MAP

So far, we have discussed linear regression in terms of maximum likelihood. But, as before, we can also propose a MAP objective. This means we specify a prior over $\mathbf{w}$. In particular, we select a prior to help *regularize* overfitting to the observed data. We will discuss two common priors (regularizers): the Gaussian prior (ℓ_2 norm) and the Laplace prior (ℓ_1 norm), shown in Figure 10.1.

Let's start with the Gaussian prior. We assume each element w_j has a Gaussian prior $\mathcal{N}(0, \sigma^2/\lambda)$, with zero covariance between the weights, for some $\lambda > 0$ and under the assumption that $p(y|\mathbf{x}) = \mathcal{N}(\mathbf{x}^\top \mathbf{w}, \sigma^2)$. The choice of the constant σ^2/λ for the prior variance is explained below. By picking a Gaussian on each w_j, we get the prior $p(\mathbf{w}) = p(w_1)p(w_2)\dots p(w_d)$. Taking the log of this zero-mean Gaussian prior, we get

$$\begin{aligned}
-\ln p(\mathbf{w}) &= -\sum_{j=1}^{d} \ln p(w_j) = -\sum_{j=1}^{d} \ln \left(\tfrac{1}{\sqrt{2\pi\sigma^2/\lambda}} \exp(-\frac{w_j^2}{2\sigma^2/\lambda} \right) \\
&= -\sum_{j=1}^{d} -\tfrac{1}{2} \ln(2\pi\sigma^2/\lambda) - \frac{w_j^2}{2\sigma^2/\lambda} \\
&= \tfrac{d}{2} \ln(2\pi\sigma^2/\lambda)) + \frac{\lambda}{2\sigma^2} \sum_{j=1}^{d} w_j^2
\end{aligned}$$

We can drop the first term, which does not affect the selection of $\mathbf{w}$ since it is constant. We can combine the negative log-likelihood and the negative log prior. Then ignoring constants, we can add up the negative log-likelihood and negative log prior to get

$$\begin{aligned}
\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} -\ln(p(\mathbf{y}|\mathbf{X}, \mathbf{w})) - \ln p(\mathbf{w}) &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(\mathbf{x}_i^\top \mathbf{w} - y_i \right)^2 + \frac{\lambda}{2\sigma^2} \sum_{j=1}^{d} w_j^2 \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} \tfrac{1}{2} \sum_{i=1}^{n} \left(\mathbf{x}_i^\top \mathbf{w} - y_i \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2.
\end{aligned}$$

Recall that $\mathbf{x}_i^\top \mathbf{w} = \sum_{j=0}^{d} w_j x_{ij}$, which gives the prediction $\hat{y}_i$. Notice that the regularization does not include w_0, because the intercept term only shifts the function. It does not

increase the complexity of the function, and so does not notably contribute to overfitting. It is preferable to avoid regularizing w_0, so that it can accurately learn the mean value of the target across $\mathbf{x}$.

Exercise 22: Show that the learned $w_0 = \frac{1}{n}\sum_{i=1}^{n} y_i$, if we first normalize the data to have zero mean, i.e. $\sum_{i=1}^{n} x_{i,j} = 0$ for every $j = 1, 2, \ldots, d$. Notice that w_0 is approximating $\mathbb{E}[Y]$. You can use this to conclude that if y_i is centered across all samples, then we did not need to add an intercept term (i.e., $w_0 = 0$). Centering involves taking the average values across samples, and subtracting it from each point: $\tilde{y}_i = y_i - \frac{1}{n}\sum_{i=1}^{n} y_i$. □

In summary, if we assume that each weight, except w_0, has a zero-mean Gaussian prior $\mathcal{N}(0, \lambda^{-1}\sigma^2)$, then we get the following ℓ_2-regularized problem[1], also called ridge regression:

$$c(\mathbf{w}) = \tfrac{1}{2}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2 \tag{10.1}$$

where λ is a user-selected parameter that is called the *regularization parameter*. The idea is to penalize weight coefficients that are too large; the larger the λ, the more large weights are penalized. Correspondingly, larger λ corresponds to a smaller covariance in the prior, pushing the weights to stay near zero. The MAP estimate, therefore, has to balance between this prior on the weights, and fitting the observed data.

Similarly to linear regression, we can take the gradient of this objective to get a system of $d + 1$ equations. We can obtain a closed form solution, but will use stochastic gradient descent. Instead of using cumulative errors, we use an average error by normalizing by n:

$$c(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2 \tag{10.2}$$

Dividing both terms by n does not change the objective, it simply rescales it. The form in Equation (10.2) is more amenable for our stochastic gradient descent solution approach, which is why we pick that instead of (10.1). We can write this objective as an average of $c_1, c_2, \ldots, c_n$ using $c_i(\mathbf{w}) = \frac{1}{2}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2$ because

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n}\left[\tfrac{1}{2}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2\right] \\
&= \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2 \\
&= \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2 = c(\mathbf{w})
\end{aligned}$$

This form makes it more clear that regularization diminishes with a growing number of samples; in other words, the prior is washed away with more data. The gradient for each term c_i is composed of the following partial derivatives for $j \in \{1, 2, \ldots, d\}$

$$\frac{\partial c_i(\mathbf{w})}{\partial w_j} = \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right) x_{ij} + \tfrac{\lambda}{n} w_j \qquad \text{and for } j = 0 \;\; \frac{\partial c_i(\mathbf{w})}{\partial w_0} = \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)$$

[1] It is called ℓ_2-regularized linear regression, because the regularizer uses an ℓ_2-norm on the weights. See the notation at the beginning of these notes, for the definitions of norms on vectors.

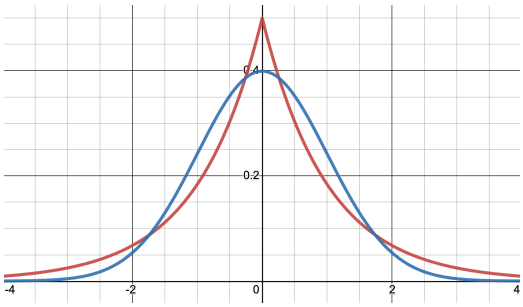


Figure 10.1: A comparison between Gaussian (blue) and Laplace (red) priors. The blue curve is $(2\pi)^{-1/2}\exp(-x^2/2)$, which is the pdf for a $\mathcal{N}(0,1)$. The red curve is $(1/2)\exp(-|x|)$, which is the pdf for a Laplace with mean zero and $b = 1$. Both prefer values to be near zero, but the Laplace prior more strongly prefers the values to equal zero.

because for $\tilde{\lambda} = \lambda/n$

$$\frac{\partial \frac{\tilde{\lambda}}{2}\sum_{k=1}^{d} w_k^2}{\partial w_j} = \frac{\tilde{\lambda}}{2}\sum_{k=1}^{d}\frac{\partial w_k^2}{\partial w_j} = \frac{\tilde{\lambda}}{2}\frac{\partial w_j^2}{\partial w_j} = \tilde{\lambda} w_j.$$

Each stochastic gradient descent update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$$

where $\mathbf{g}_t = \nabla c_i(\mathbf{w})$ for a random sample i, with

$$\nabla c_i(\mathbf{w}) = \begin{bmatrix} (\mathbf{x}_i^\top \mathbf{w} - y_i) \\ (\mathbf{x}_i^\top \mathbf{w} - y_i)\, x_{i1} + \lambda w_1 \\ (\mathbf{x}_i^\top \mathbf{w} - y_i)\, x_{i2} + \lambda w_2 \\ \vdots \\ (\mathbf{x}_i^\top \mathbf{w} - y_i)\, x_{id} + \lambda w_d \end{bmatrix}$$

Exercise 23: Write down the mini-batch gradient descent update. □

We can go through the same procedure with a different prior: a Laplace prior. If we go through similar steps as above, we get an ℓ_1 penalized objective

$$c(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 + \frac{\lambda}{n}\sum_{j=1}^{d}|w_j| \tag{10.3}$$

which is often called the Lasso. We put a Laplace prior on each weight $p(w_j) = \frac{1}{2b}\exp(-\frac{|w_j-\mu|}{b})$ with parameters $\mu = 0$ and scale $b = \sigma^2/\lambda$. This results in putting an ℓ_1 regularizer on the weights, which sums up the absolute values of the weights.

Exercise 24: Derive Equation (10.3) using the MAP formulation with the given Laplace prior, similarly to how it was done for MAP with a Gaussian prior on the weights. □

As with the ℓ_2 regularizer for ridge regression, the ℓ_1 regularizer penalizes large values in $\mathbf{w}$. However, it also produces more sparse solutions, where entries in $\mathbf{w}$ are zero. This preference can be seen in Figure 10.1, where the Laplace distribution is more concentrated around zero. In practice, however, this preference is even stronger than implied by the distribution, due to how the spherical least-squares loss and the ℓ_1 regularizer interact.

Forcing entries in $\mathbf{w}$ to zero has the effect of feature selection, because zeroing entries in $\mathbf{w}$ is equivalent to removing the corresponding feature. Consider the dot product

$$\mathbf{x}^\top \mathbf{w} = \sum_{j=0}^{d} x_j w_j = \sum_{j:w_j \neq 0} x_j w_j.$$

This is equivalent to simply dropping entries in $\mathbf{x}$ and $\mathbf{w}$ where $w_j = 0$. Notice that again it is not sensible to apply this regularizer to w_0, and it remains unregularized.

For the Lasso, we no longer have a closed-form solution. We do not have a closed form solution, because we cannot solve for $\mathbf{w}$ in closed-form that provides a stationary point. Instead, we use gradient descent to compute a solution to $\mathbf{w}$. The ℓ_1 regularizer, however, is non-differentiable at 0. Understanding how to optimize this objective requires a bit more optimization background; we leave it for a future course.

10.2 Expectation and Variance for the Regression Solutions

A natural question to ask is how this regularization parameter can be selected, and the impact on the final solution vector. The selection of this regularization parameter leads to a bias-variance trade-off. To understand this trade-off, we need to understand what it means for the solution to be biased, and how to characterize the variance of the solution, across possible datasets.

Let us begin with understanding the bias and variance of the non-regularized solution. For simplicity in the derivation, let's look only at the univariate setting: input $x \in \mathbb{R}$ and weights $w \in \mathbb{R}$, with $f(x) = xw$. For this analysis, we start by presuming the distributional assumptions behind linear regression are true. This means that there exists a true parameter ω such that for each of the data points

$$Y_i = \omega X_i + \varepsilon_i$$

where the ε_j are i.i.d. random variables drawn according to $\mathcal{N}(0, \sigma^2)$. We can characterize the MLE weights (estimator) w_{MLE} as a random variable, where the randomness is across possible datasets that could have been observed. In this sense, we are considering the dataset $\mathcal{D}$ to be a random variable, and the solution $w_{\text{MLE}}(\mathcal{D})$ from that dataset as a function of this random variable. All stochasticity comes from the fact that we could have drawn different datasets $\mathcal{D}$, and we want to reason about the resulting distribution over possible $w_{\text{MLE}}(\mathcal{D})$. Our learned estimator will be just one of these possible solutions, but we reason about the whole space to help us understand properties of our solution.

We can show that $w_{\text{MLE}}(\mathcal{D})$ is an unbiased estimator of ω. To do so, we will need the closed form solution, which we actually already derived in Equation (5.5):

$$w_{\text{MLE}}(\mathcal{D}) = \frac{\sum_{i=1}^{n} X_i Y_i}{\sum_{i=1}^{n} X_i^2} \tag{10.4}$$

For simplicity of notation, we will use $S_n \stackrel{\text{def}}{=} \sum_{i=1}^n X_i^2$. Then we have that

$$\begin{aligned}
\mathbb{E}[w_{\text{MLE}}(\mathcal{D})] &= \mathbb{E}\left[\frac{\sum_{i=1}^n X_i Y_i}{S_n}\right] \\
&= \mathbb{E}\left[\frac{\sum_{i=1}^n X_i(\omega X_i + \varepsilon_i)}{S_n}\right] && \triangleright \text{ because } Y_i = \omega X_i + \varepsilon_i \\
&= \mathbb{E}\left[\frac{\sum_{i=1}^n (\omega X_i^2 + X_i\varepsilon_i)}{S_n}\right] \\
&= \mathbb{E}\left[\frac{\omega \sum_{i=1}^n X_i^2 + \sum_{i=1}^n X_i\varepsilon_i}{S_n}\right] \\
&= \mathbb{E}\left[\frac{\omega S_n}{S_n}\right] + \mathbb{E}\left[\frac{\sum_{i=1}^n X_i\varepsilon_i}{S_n}\right] && \triangleright \text{ by linearity of expectation} \\
&= \mathbb{E}\left[\omega\right] + \sum_{i=1}^n \mathbb{E}\left[\varepsilon_i \frac{X_i}{S_n}\right] \\
&= \omega + \sum_{i=1}^n \mathbb{E}[\varepsilon_i]\mathbb{E}\left[\frac{X_i}{S_n}\right] && \triangleright \ \omega \text{ not random and } \varepsilon_i \text{ independent of all } X_i \\
&= \omega && \triangleright \text{ because } \mathbb{E}\left[\varepsilon_i\right] = 0
\end{aligned}$$

Therefore, we can conclude that $\mathbb{E}[w_{\text{MLE}}(\mathcal{D})] = \omega$ and so $w_{\text{MLE}}(\mathcal{D})$ is an unbiased estimator.

We can similarly characterize the variance, and obtain

$$\begin{aligned}
\text{Var}[w_{\text{MLE}}(\mathcal{D})] &= \mathbb{E}\left[\left(w_{\text{MLE}}(\mathcal{D}) - \omega\right)^2\right] \\
&= \mathbb{E}\left[w_{\text{MLE}}(\mathcal{D})^2\right] - \omega^2
\end{aligned}$$

Above we showed that $w_{\text{MLE}}(\mathcal{D}) = \omega + \sum_{i=1}^n \varepsilon_i \frac{X_i}{S_n}$. Again, for simplicity of notation, define this residual term $R_n \stackrel{\text{def}}{=} \sum_{i=1}^n \varepsilon_i \frac{X_i}{S_n}$. Then we get

$$\begin{aligned}
w_{\text{MLE}}(\mathcal{D})^2 &= (\omega + R_n)^2 \\
&= \omega^2 + 2\omega R_n + R_n^2
\end{aligned}$$

We can further show that

$$\mathbb{E}[\omega R_n] = \omega \mathbb{E}[R_n] = \omega \sum_{i=1}^n \mathbb{E}[\varepsilon_i]\mathbb{E}\left[\frac{X_i}{S_n}\right] = 0$$

as we showed above when characterizing the expectation of $w_{\text{MLE}}(\mathcal{D})$. We can use the law of total probability to show that

$$\mathbb{E}[R_n^2] = \sigma^2 \mathbb{E}[S_n^{-1}]$$

We leave this as an exercise. Putting this all together, we get that

$$\begin{aligned}
\text{Var}[w_{\text{MLE}}(\mathcal{D})] &= \mathbb{E}\left[w_{\text{MLE}}(\mathcal{D})^2\right] - \omega^2 \\
&= \omega^2 + 0 + \mathbb{E}[R_n^2] - \omega^2 \\
&= \sigma^2 \mathbb{E}[S_n^{-1}] \\
&= \sigma^2 \mathbb{E}\left[\frac{1}{n} C_n^{-1}\right] && \triangleright \text{ for } C_n \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n X_i^2
\end{aligned}$$

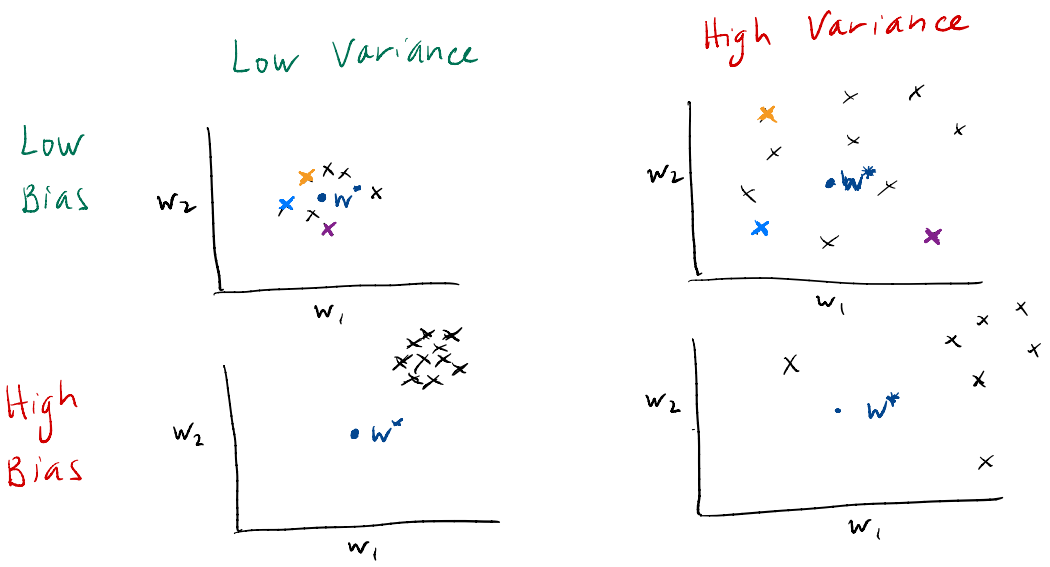


Figure 10.2: Visualizing the meaning of bias and variance for our regression estimates. This visualization is for a setting with two weights, $\mathbf{w} = (w_1, w_2)$, where each $\times$ corresponds to possible $\mathbf{w}$ we might see for a given dataset. For example, the purple $\times$ would be the weights learned under a randomly sampled dataset $\mathcal{D}_1$; the orange $\times$ the weights under a different randomly sampled dataset $\mathcal{D}_2$; and the light blue $\times$ the weights under a third randomly sampled dataset $\mathcal{D}_3$. All these datasets are possible datasets that could have been observed, although we of course only see one dataset and our $\times$ is one amongst these many possibilities. ***Those in the first row with low bias*** *have all the possible solutions around the true $\mathbf{w}^*$, where for low variance they are clustered more closely around $\mathbf{w}^*$ and for high variance any one $\times$ can be quite far. The second row visualizes a situation where there is high bias, and the solutions $\times$ are clustered away from the true $\mathbf{w}^*$.*

The last format is given in terms of the sample average estimate C_n, which is essentially a sample average estimate of the variance for X_i. (It would be a variance estimate, if it was centered around the mean.) This term reflects the variability in X. For a small number of samples, C_n could vary widely, and could be very small. The inverse in the above can be very big, and so the variance for $w_{\text{MLE}}(\mathcal{D})$ can be big for a small amount of data. This implies that, across datasets, the solution $w_{\text{MLE}}(\mathcal{D})$ can vary widely. This behavior is not desirable: if our solution could be very different across several different random subsets of data, we cannot be confident in any one of these solutions. Notice that as n get bigger, the variance decreases proportionally to n^{-1}, because of the $\frac{1}{n}$ in front of C_n. We depict this in Figure 10.2.

The regularized solution, on the other hand, is much less likely to have high variance, but will no longer be unbiased. Let $w_{\text{MAP}}(\mathcal{D})$ be the MAP estimate for the ℓ_2 regularized problem with some $\lambda > 0$. As above, we can write the MAP estimate as a closed form solution

$$w_{\text{MAP}}(\mathcal{D}) = \frac{\sum_{i=1}^{n} X_i Y_i}{\lambda + S_n} \tag{10.5}$$

Then, using similar steps to above we get

$$\begin{aligned}\mathbb{E}[w_{\text{MAP}}(\mathcal{D})] &= \mathbb{E}\left[\frac{\sum_{i=1}^n X_i Y_i}{\lambda + S_n}\right] \\ &= \mathbb{E}\left[\frac{\omega S_n}{\lambda + S_n}\right] + \mathbb{E}\left[\frac{\sum_{i=1}^n X_i \varepsilon_i}{\lambda + S_n}\right] \\ &= \omega \mathbb{E}\left[\frac{S_n}{\lambda + S_n}\right] + 0 \\ &\neq \omega\end{aligned}$$

when $\lambda > 0$. Notice that if $\lambda = 0$, then $w_{\text{MAP}}(\mathcal{D})$ is unbiased. In fact, it simply correspond to the MLE solution for $\lambda = 0$, so it makes sense that it is unbiased for $\lambda = 0$. The bias is determined by how far $\mathbb{E}\left[\frac{S_n}{\lambda+S_n}\right]$ is from 1. As $\lambda \to 0$, the solution becomes less and less biased. As $\lambda \to \infty$, the solution becomes maximally biased with $\mathbb{E}\left[\frac{S_n}{\lambda+S_n}\right] \to 0$.

We can also characterize the variance of $w_{\text{MAP}}(\mathcal{D})$. We leave the steps as an exercise. Again, for simplicity of notation, let's define $C_{n,\lambda} = \frac{1}{n}(\lambda + S_n)$. Then variance is

$$\text{Var}[w_{\text{MAP}}(\mathcal{D})] = \sigma^2 \mathbb{E}\left[\frac{1}{n} C_{n,\lambda}^{-1} C_n C_{n,\lambda}^{-1}\right]$$

Notice now that even if C_n is very small, it does not cause the variance to become very big because we are not inverting it. Instead, we use the inverse of $C_{n,\lambda}$. This inverse is always smaller than $1/\lambda$, because we increase the denominator by λ. Therefore, for reasonably large λ, the variance will not be very big and it should be notably smaller than $w_{\text{MLE}}(\mathcal{D})$.

We can reason about which quadrants best characterize our regression solutions. The w_{MLE} solution is unbiased, so it should be in the first row. For a small number of samples, the variance is likely high, and so it would be in the low-bias, high-variance quadrant. Once we have enough samples, the variance becomes small—because it decreases proportionally to n—and so w_{MLE} is in the low-bias, low-variance quadrant.

The w_{MAP} solution is biased, so we have to think a bit more carefully. For a very small λ, we know that the bias is minimal, so the solution will be low-bias. So, this answer depends on the choice of λ. For a small λ, we expect w_{MAP} to behave somewhat similarly to w_{MLE}, but should be lower-variance. For a larger λ, the bias is higher and the variance even lower, but this bias also decreases with samples. In general, for a large λ with a small number of samples, we except a high-bias, low-variance solution. As we get more samples, we start to decrease the bias until with enough samples we have a low-bias, low-variance solution.

10.3 The Bias-Variance Trade-off

The reason we care about the bias and variance is that the expected mean-squared error to the true weights can be decomposed into the bias and variance. We saw this in Section 3.5, when we talked about sample average estimators. As depicted in Figure 10.3, there is an optimal choice of λ that minimizes this bias-variance trade-off—if we could find it. We can show the same decomposition for the weights for regression

$$\mathbb{E}\left[\|\mathbf{w}(\mathcal{D}) - \boldsymbol{\omega}\|_2^2\right] = \mathbb{E}\left[\sum_{j=1}^d (w_j(\mathcal{D}) - \omega_j)^2\right] = \sum_{j=1}^d \mathbb{E}\left[(w_j(\mathcal{D}) - \omega_j)^2\right]$$

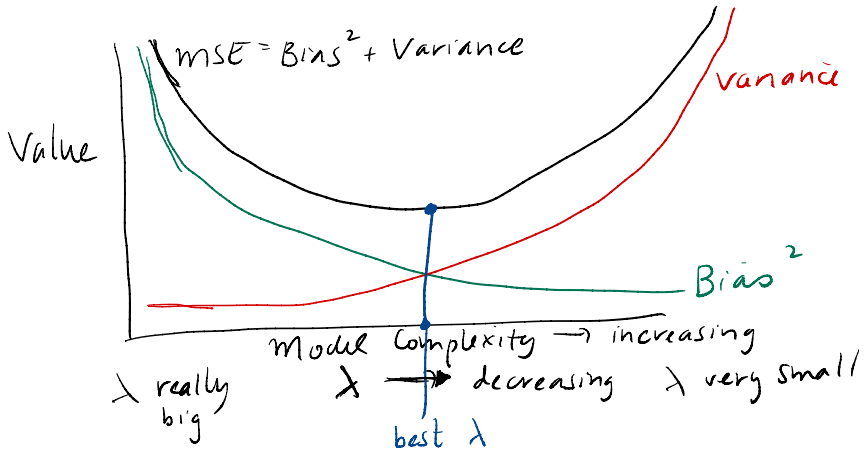


Figure 10.3: The bias-variance trade-off. The Model Complexity is increasing on the x-axis. For example, increasing p would correspond to increasing model complexity. For the regularization parameter, a smaller λ constrains the function less, and so it corresponds to higher model complexity. A large λ constraints the weights much more, to be near zero, and so reduces the model complexity.

where we can then further simplify this inner term

$$\begin{aligned}\mathbb{E}\left[(w_j(\mathcal{D}) - \omega_j)^2\right] &= \mathbb{E}\left[(w_j(\mathcal{D}) - \mathbb{E}\left[w_j(\mathcal{D})\right] + \mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)^2\right] \\ &= \mathbb{E}\left[(w_j(\mathcal{D}) - \mathbb{E}\left[w_j(\mathcal{D})\right])^2\right] + \mathbb{E}\left[(\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)^2\right]\end{aligned}$$

where the second step follows from the fact that

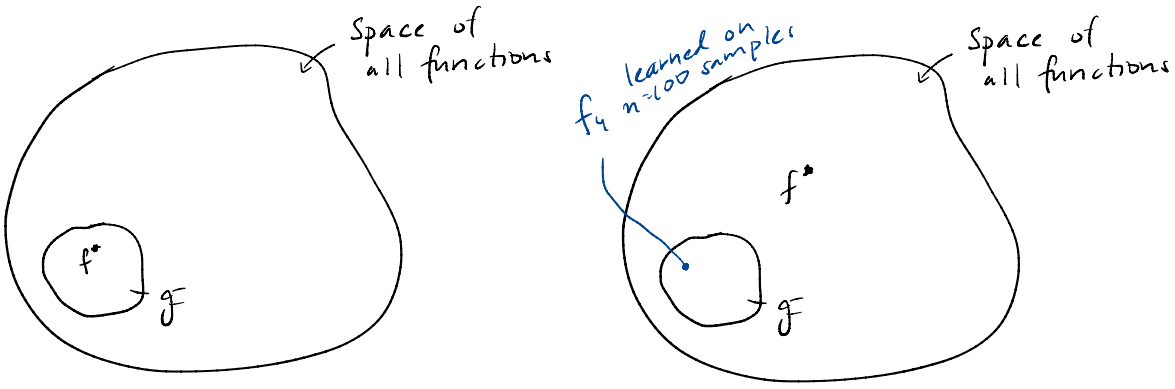
$$\begin{aligned}-2\mathbb{E}\left[(w_j(\mathcal{D}) - \mathbb{E}\left[w_j(\mathcal{D})\right])(\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)\right] &= (\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)\mathbb{E}\left[w_j(\mathcal{D}) - \mathbb{E}\left[w_j(\mathcal{D})\right]\right] \\ &= 0.\end{aligned}$$

The first term above in $\mathbb{E}\left[(w_j(\mathcal{D}) - \omega_j)^2\right]$ is the variance of the jth weight and the second term is the bias of the jth weight, where $\mathbb{E}\left[(\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)^2\right] = (\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)^2$ because nothing is random in this term so the outer expectation is dropped. This gives

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{w}(\mathcal{D}) - \boldsymbol{\omega}\|_2^2\right] &= \sum_{j=1}^{d} \mathbb{E}\left[(w_j(\mathcal{D}) - \omega_j)^2\right] \\ &= \sum_{j=1}^{d} (\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j)^2 + \mathrm{Var}\left[w_j(\mathcal{D})\right]\end{aligned}$$

showing that the expected mean-squared error to the true weight vector $\boldsymbol{\omega}$ decomposes into the squared bias $\mathbb{E}\left[w_j(\mathcal{D})\right] - \omega_j$ and the variance $\mathrm{Var}\left[w_j(\mathcal{D})\right]$. The bias-variance trade-off reflects the fact that we could potentially reduce the mean-squared error by incurring some bias, as long as the variance is decreased more than the squared bias.

Above we assumed that the true model was linear, and so the only bias introduced was from the regularization. This was the *realizable* setting, where our function class contains



(a) Realizable setting: $f^* \in \mathcal{F}$ *(b) Non-realizable setting: $f^* \notin \mathcal{F}$*

Figure 10.4: Visualizing the realizable *and* non-realizable *settings. If f^* is representable by our function class, then it is the realizable setting. For example, if $\mathcal{F}$ is all degree-4 polynomials, and f^* is a polynomial with $p = 3$, then $f^* \in \mathcal{F}$ and we are in the realizable setting. On the other hand, if f^* is a polynomial with $p = 6$, then $f^* \notin \mathcal{F}$ and we are in the non-realizable setting. For the realizable setting, it does not mean we will find f^* with polynomial regression on a given (finite) dataset. But, in the limit as we get more n, polynomial regression will find this f^*. For the non-realizable setting, in the limit we will find the best approximation to f^*. Of course, for either case, any function we learn will be in $\mathcal{F}$. For this example, with $\mathcal{F}$ the space of all degree-4 polynomials, we have labeled our learned function f_4 in the diagram, since it is a degree four polynomial.*

the true function. Specifically, we assumed that the true function is linear, and that the bias introduced was only due to regularization. In reality, when using linear regression with regularization, we are introducing bias both from

1. selecting a simpler function class, and
2. from the regularization.

For example, we might select $\mathcal{F}$ to be the set of linear functions, the true function f^* might be cubic function. We visualize this in Figure 10.4.

If the true function is not linear, then we cannot compare the learned weights for a linear function directly to the true function. If a powerful basis is used to first transform the data, then we can learn nonlinear functions even though the solution uses linear regression. In this case, it is feasible that this function class is sufficiently powerful and includes the true function, and that the bias is mostly due to regularization. But, in general, it will be difficult to guarantee that we have specified a function class that includes the true function, and it will be difficult to directly compare our parameters to true parameters (which may not even be of the same dimension).

We can more generally talk about bias and variance by considering instead the reducible error. In fact, the bias-variance trade-off is all about reducing the reducible error. (Remember, we cannot reduce the irreducible error—the name says it all—by improving how we estimate the function.) We can define a more general bias-variance decomposition that

compares function outputs rather than parameter vectors. Recall the reducible error corresponds to $\mathbb{E}\left[(f_{\mathcal{D}}(\boldsymbol{X}) - f(\boldsymbol{X}))^2\right]$, where $f(\boldsymbol{X})$ is the optimal function, i.e., $f(\mathbf{x}) = \mathbb{E}\left[Y|\mathbf{x}\right]$ for the squared cost. We previously discussed this reducible error for a fixed function, with expectation only over $\boldsymbol{X}$. But now we additionally consider the fact that $f_{\mathcal{D}}$ is random, and we can reason about its expectation and variance for a given $\mathbf{x}$.

Let's start by only considering the expected mean-squared error, for a given input $\mathbf{x}$. Using similar steps to the decomposition above, we get

$$\mathbb{E}\left[(f_{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2)\right] = \left(\mathbb{E}\left[f_{\mathcal{D}}(\mathbf{x})\right] - f(\mathbf{x})\right)^2 + \mathrm{Var}\left[f_{\mathcal{D}}(\mathbf{x})\right].$$

Notice that in the second line, the expectation is now inside the squared distance; this term corresponds to the squared bias. The bias here reflects the output of the estimated function $f_{\mathcal{D}}(\mathbf{x})$, in expectation across all datasets $\mathcal{D}$. The variance term reflects how much the prediction for $\mathbf{x}$ can vary, if we learn on different iid datasets. This decomposition of the mean-squared error into a squared bias and variance is not obvious, but does follow similar steps to above. It is left as an exercise.

The above generalization highlights that one of the ways we balance bias and variance is actually in the selection of the function class. If we select a simple function class, the class is likely not large enough—not powerful enough—to represent the true function. This introduces some bias, but likely also has lower variance, because that simpler function class is less likely to overfit to any one dataset. If this class is too simple, then we might be suffering from underfitting. On the other hand, if we select a more powerful function class, that does contain the true function, we may not have any bias but could have high variance due to the ability to find a function in your large class that overfits a given dataset. Though we have the ability to learn a highly accurate function, it will be difficult to actually find that function amongst this larger class. Instead, one is likely to select a model that overfits to the given data, and does not generalize to new data (i.e., performs poorly on new data).

We can revisit our bias-variance quadrants and categorize these learning scenarios.

1. **Low-Bias, Low-Variance**: (a) Large $\mathcal{F}$ (high model complexity) so that we can nearly represent f^*, with a very big n. (b) Small $\mathcal{F}$ (low model complexity) but f^* is also simple, such as the case where $\mathcal{F}$ is composed of linear functions and f^* is also linear. For this setting, we do not need as many samples, for even for a relatively small n we might have low variance.

2. **Low-Bias, High-Variance**: Large $\mathcal{F}$ (high model complexity) so that we can nearly represent f^* (low-bias), but$n$ is not big enough.

3. **High-Bias, Low-Variance**: Small $\mathcal{F}$ (low model complexity) but f^* is complex, such as the case where $\mathcal{F}$ is composed of linear functions and f^* is a high-order polynomial, say with $p = 8$.

4. **High-Bias, High-Variance**: This is a bad quadrant to be in. It could occur if we have very little data and interim model complexity. For example, $\mathcal{F}$ could be composed of cubic functions, which cannot represent f^* (a polynomial with $p = 8$), and n is small enough that there is large variability amongst possible cubic functions that could be observed.

Exercise 25: In which four quadrants is the training accuracy reflective of generalization error? In which four quadrants is the generalization error low or high? □

10.4 Selecting Models for Deployment

Finding the balance between bias and variance, and between underfitting and overfitting, is a core problem in machine learning. Our goal is to identify the true function, and in some cases the data may be insufficient for identification. For example, imagine you are given a dataset of images, where the color red has no impact on prediction accuracy. Your classifier, though, does not know that this property is irrelevant and may use it to better fit the data. If there were multiple instances of the same picture, with and without the color red, it might be able to learn that that property is not relevant. But that is too much to hope for. Instead, we build-in some prior knowledge into the types of functions we learn to acknowledge that the given data is unlikely to be sufficient to perfectly identify the model. That prior knowledge need not be specific to the problem; it could be as simple as preferring to use a minimal set of features in the observed data. Understanding what this prior knowledge should be—understanding inductive biases—remains an important question.

However, what you have learned so far already gives you some tools to answer this question in practice. We have two strategies to select models: using first principles (based on reasoning about the bias and variance) and using model evaluation. Notice that we do not directly optimize the bias-variance trade-off. We cannot actually measure the bias, so we do not directly minimize these terms. Rather, this decomposition guides how we select models. We may reason that if we have a small dataset, then we should err on the side of using simpler models. This might mean we choose to use a smaller p for polynomial regression, or incorporate regularization.

Such reasoning helps us constrain the models we consider; then testing lets us select amongst this smaller set. We can take our dataset and separate it into a training set and a test set. Because we are using this test set to select amongst different models, it is typically called a *validation set*. We might train three different models, with $p = 2, 3, 4$, and then evaluate performance on the validation set, selecting the best p. We then select that p, and train on all the data before deploying.

Note that in some cases there is actually a training set, validation set and test set. To be more sure about the model before deploying, we might actually split the dataset into training and test first, then further using this validation approach with another split on the training set. The model can be tested once more on the test set, before deployment, to ensure we are deploying an acceptable model. Note that before testing the model on the test set, we do still combine the training and validation into one dataset, and learn that model on this larger dataset, before testing on the test set.

Exercise 26: A test set allows us to measure the accuracy of our predictor. Imagine we have trained a predictor using polynomial regression with $p = 4$, selected using validation. We measure the mean squared error on the test dataset. We want to ensure we are not tricked by an optimistic average error. How can we say that, with 95% confidence, our predictor will not do worse than some lower bound? □

Chapter 11

Logistic Regression and Linear Classifiers

In Chapter 7, when introducing prediction, we presented a classifier as a function $f : \mathcal{X} \to \mathcal{Y}$ for a finite (unordered) set $\mathcal{Y}$ and showed that a reasonable goal to obtain a good classifier is to approximate $p(y|\mathbf{x})$. Similarly to linear regression, we need to figure out how to parameterize $p(y|\mathbf{x})$. Here we consider the simplest first case: $\mathcal{Y} = \{0, 1\}$ (binary classification). We know $p(y|\mathbf{x})$ must be a (conditional) Bernoulli distribution, because Y is a binary variable. The parameter for a Bernoulli distribution is $\alpha(\mathbf{x}) = p(y = 1|\mathbf{x})$, the success probability. In this chapter, we discuss how to parameterize and learn this $\alpha(\mathbf{x})$, with an approach called *logistic regression*.

11.1 The Parameterization for Binary Classification

Let us start by reasoning about how to represent $\alpha(\mathbf{x}) = p(y = 1|\mathbf{x})$. We could again use a linear function of the inputs $\mathbf{x}$. Notice, however, that a linear function $\mathbf{x}^\top \mathbf{w}$ may produce any number in $\mathbb{R}$. We need to approximate $p(y = 1|\mathbf{x})$, which has to be between 0 and 1. The simple idea is to still use a simple linear function of $\mathbf{x}$, but then squash the values between 0 and 1 with what is called the sigmoid function, shown in Figure 11.1. We approximate $p(y = 1|\mathbf{x})$ with

$$\sigma(\mathbf{x}^\top \mathbf{w}) = \left(1 + \exp(-\mathbf{x}^\top \mathbf{w})\right)^{-1}.$$

The Bernoulli distribution over Y, with α a function of $\mathbf{x}$, is

$$p(y|\mathbf{x}) = \begin{cases} \left(\frac{1}{1+e^{-\boldsymbol{\omega}^\top \mathbf{x}}}\right) & \text{for } y = 1 \\ \left(1 - \frac{1}{1+e^{-\boldsymbol{\omega}^\top \mathbf{x}}}\right) & \text{for } y = 0 \end{cases} \tag{11.1}$$
$$= \sigma(\mathbf{x}^\top \mathbf{w})^y (1 - \sigma(\mathbf{x}^\top \mathbf{w}))^{1-y}$$

In the realizable setting, we assume there is are true underlying parameters $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_d)$ that satisfy

$$p(y = 1|\mathbf{x}) = \sigma(\boldsymbol{\omega}^\top \mathbf{x}).$$

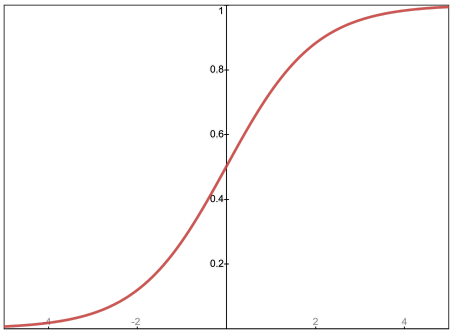


Figure 11.1: Sigmoid function $\sigma(t) = \frac{1}{1+\exp(-t)}$ *for* $t \in [-5, 5]$.

Our goal is to identify these parameters. Notice that this means that our goal is to predict the probability that the class is 1; given this probability, we can infer $p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x})$. In the next section, we talk about how to learn $\mathbf{w}$; here, we first discuss a bit more how we use this model.

The function learned by logistic regression returns a probability, rather than an explicit prediction of 0 or 1. Therefore, we have to take this probability estimate and convert it to a suitable prediction of the class. For a previously unseen data point $\mathbf{x}$ and a set of learned coefficients $\mathbf{w}$, we simply calculate the conditional probability as

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{x}^\top \mathbf{w}}}.$$

If $p(y = 1|\mathbf{x}, \mathbf{w}) \geq 0.5$ we conclude that data point $\mathbf{x}$ should be labeled as positive ($\hat{y} = 1$). Otherwise, if $p(y = 1|\mathbf{x}, \mathbf{w}^*) < 0.5$, we label the data point as negative ($\hat{y} = 0$). The predictor maps a $(d + 1)$-dimensional vector $\mathbf{x} = (x_0 = 1, x_1, \ldots, x_d)$ into a zero or one.

Notice that, even if logistic regression can perfectly model $p(y = 1|\mathbf{x})$, this does not mean we obtain perfect classification accuracy. Imagine you were given the true $p(y = 1|\mathbf{x})$ that generates the data. Imagine for one observation vector, $p(y = 1|\mathbf{x}) = 0.5$. This means that, for this given observation, 50% of the time is labeled positive and 50% it is labeled negative. This goes back to partial observability and irreducible error. The given observations are insufficient to perfectly characterize the outcome. Potentially, if we had obtained a richer observation vector $\mathbf{x}$ with more information, the target y might become more certain and the distribution over y more concentrated at one value. But, we are stuck with the data we have been given, and so have to recognize that sometimes a class label is simply ambiguous, even under the optimal model.

The probability values themselves can be useful. If the probability estimates are accurate, then they provide a measure of confidence in the classification. You might be more comfortable making a health decision if the classifier $p(y = 1|\mathbf{x}) = 0.99$ rather than if $p(y = 1|\mathbf{x}) = 0.6$. Additionally, differences in probability estimates can help you pick between classifiers. For example, if you have two classifiers that produce good classification accuracy on a test set, then it is preferable to have a classifier that consistently produces probabilities near 0.9 and 0.1, rather than probabilities that hover around 0.5. The reason for this is that small perturbations are expected to have more impact on the second classifier, which could suddenly erroneously swap the labeling on an instance.

Remark: As we discussed in Chapter 9, the threshold for classification need not be 0.5. In some cases, one might care more about failing to identify a positive (e.g., failing to identify a disease); in such a case, it may be safer to err on the side of a smaller threshold, so that more instances are labeled as positive. For now, we will assume this simpler thresholding, but remain cognizant that the choice can be an important one.

11.2 Maximum Likelihood for Logistic Regression

To learn the parameters for logistic regression, we need to define an objective. We will once again use maximum likelihood to help us derive a reasonable objective. As before, assume that the data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is an i.i.d. sample from a fixed but unknown probability distribution $p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x})$. The data is generated by randomly drawing a point $\mathbf{x}$ according to $p(\mathbf{x})$ and then setting its class label Y according to the Bernoulli distribution in (11.1). Our objective is the negative log-likelihood for the conditional distribution, scaled by the number of samples

$$c(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} - \ln p(y_i|\mathbf{x}_i)$$

which we can write as $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$ where

$$\begin{aligned} c_i(\mathbf{w}) &= -\ln p(y_i|\mathbf{x}_i) && \triangleright\ p(y_i|\mathbf{x}_i) = \sigma(\mathbf{x}_i^\top \mathbf{w})^{y_i}(1-\sigma(\mathbf{x}_i^\top \mathbf{w}))^{1-y_i} \\ &= -\ln \sigma(\mathbf{x}_i^\top \mathbf{w})^{y_i} - \ln(1-\sigma(\mathbf{x}_i^\top \mathbf{w}))^{1-y_i} \\ &= -y_i \ln \sigma(\mathbf{x}_i^\top \mathbf{w}) - (1-y_i)\ln\left(1-\sigma(\mathbf{x}_i^\top \mathbf{w})\right) \end{aligned}$$

This objective is typically referred to as the *cross-entropy*.

From here, you could take the derivative of each component in this sum, using the chain rule for the sigmoid. For the first component, with $p_i = \sigma(\theta_i)$,

$$\begin{aligned} \frac{\partial y_i \ln \sigma(\mathbf{x}_i^\top \mathbf{w})}{\partial w_j} &= y_i \frac{\partial \ln \sigma(\theta_i)}{\partial \theta_i}\frac{\partial \theta_i}{w_j} \\ &= y_i \frac{\partial \ln p_i}{\partial p_i}\frac{\partial p_i}{\partial \theta_i}\frac{\partial \theta_i}{w_j} \\ &= y_i \frac{1}{p_i}\frac{\partial p_i}{\partial \theta_i}\frac{\partial \theta_i}{w_j} \\ &= y_i \frac{1}{p_i}\sigma(\theta_i)(1-\sigma(\theta_i))x_{ij} \\ &= y_i(1-\sigma(\theta_i))x_{ij} \end{aligned}$$

because

$$\frac{\partial \sigma(\theta_i)}{\partial \theta_i} = \sigma(\theta_i)(1-\sigma(\theta_i)).$$

You can verify this step for yourself, by explicitly plugging in the definition of σ. For the second component, following similar steps, we get

$$\frac{\partial (1-y_i)\ln(1-\sigma(\mathbf{x}_i^\top \mathbf{w}))}{\partial w_j} = (y_i - 1)\sigma(\theta_i)x_{ij}$$

Summing these together and taking the negative, we end up with the gradient $(p_i - y_i)x_{ij}$.

Exercise 27: We could have slightly rearranged the objective before taking the gradient. This would lead to another path to derive the update rule for logistic regression. You can notice first that

$$1 - \sigma(\mathbf{x}_i^\top \mathbf{w}) = 1 - \frac{1}{1+\exp(-\mathbf{x}_i^\top \mathbf{w})} = \frac{\exp(-\mathbf{x}_i^\top \mathbf{w})}{1+\exp(-\mathbf{x}_i^\top \mathbf{w})}$$

giving

$$\begin{aligned} c_i(\mathbf{w}) &= -y_i \cdot \ln\left(1+\exp(-\mathbf{x}_i^\top \mathbf{w})\right) + (1-y_i)\cdot \ln\left(\exp(-\mathbf{x}_i^\top \mathbf{w})\right) \\ &\quad - (1-y_i)\cdot \ln\left(1+\exp(-\mathbf{x}_i^\top \mathbf{w})\right) \\ &= (y_i - 1)\,\mathbf{x}_i^\top \mathbf{w} + \ln\left(\frac{1}{1+\exp(-\mathbf{x}_i^\top \mathbf{w})}\right). \end{aligned}$$

Derive the gradient of c starting from here. □

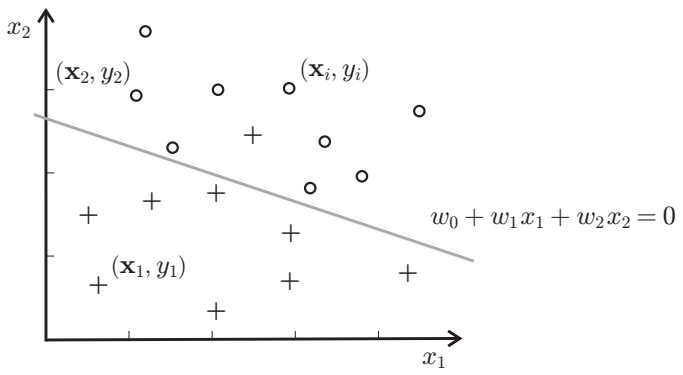


Figure 11.2: A data set in $\mathbb{R}^2$ consisting of nine positive and nine negative examples. The gray line represents a linear decision surface in $\mathbb{R}^2$. The decision surface does not perfectly separate positives from negatives.

Unlike linear regression, there is no closed-form solution to $\nabla c(\mathbf{w}) = \mathbf{0}$. Thus, we have to proceed with iterative optimization methods, like gradient descent. We initialize $\mathbf{w}_0$ usually to a random vector. Because the objective is convex, the initialization only affects the number of steps, but will not prevent the gradient descent from converging to a global minimum. The stochastic gradient descent update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\sigma(\mathbf{x}_i^\top \mathbf{w}_t) - y_i\right) \mathbf{x}_i$$

and the mini-batch gradient descent update, for mini-batch $(\mathbf{x}_k, y_k), \ldots, (\mathbf{x}_{k+b}, y_{k+b})$ is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta_t}{b} \sum_{i=k}^{k+b-1} \left(\sigma(\mathbf{x}_i^\top \mathbf{w}_t) - y_i\right) \mathbf{x}_i.$$

Exercise 28: Like linear regression, we can obtain a regularized version for logistic regression using MAP. We can incorporate a Gaussian prior—or Laplace prior—on the weights and go through the same steps as we did above for the maximum likelihood formulation. As with ridge regression, this amounts to adding the regularizer $\frac{\lambda}{2n} \sum_{j=1}^{d} w_j^2$ to the objective. The resulting update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\sigma(\mathbf{x}_i^\top \mathbf{w}_t) - y_i\right) \mathbf{x}_i - \eta_t \frac{\lambda}{n} \mathbf{w}_t.$$

Derive this MAP objective. What is the variance for the Gaussian prior, in terms of λ? □

11.3 Logistic Regression Learns a Linear Classifier

The logistic regression classifier is a linear classifier, despite the fact that the sigmoid is non-linear. This is because $p(y = 1|\mathbf{x}, \mathbf{w}) \geq 0.5$ only when $\mathbf{x}^\top \mathbf{w} \geq 0$. The expression $\mathbf{x}^\top \mathbf{w} = 0$ represents the equation of a hyperplane that separates positive and negative examples.

To better understand this, consider Figure 11.2. A linear classifier is a linear function (a point, a line, a plane or a hyperplane) that splits $\mathbb{R}^d$ into two half-spaces. The two

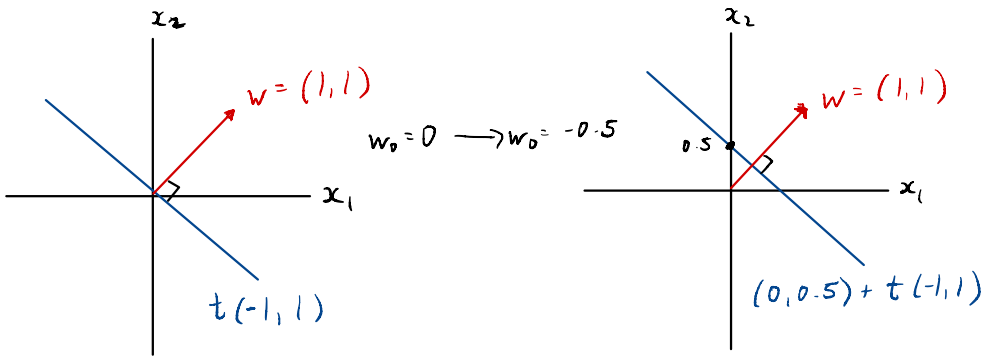


Figure 11.3: Visualizing two possible linear classifiers, one going through the origin and the other shifted away from the origin but defined by the same (w_1, w_2).

half-spaces act as decision regions for the positive and negative examples, respectively. For a given $\mathbf{x}$, if $\mathbf{x}^\top \mathbf{w} > 0$, then $\mathbf{x}$ is classified as a positive. If $\mathbf{x}^\top \mathbf{w} < 0$, then $\mathbf{x}$ is classified as a negative. It may not be possibly to perfectly separate the points; the goal is to the find the surface that best separates the points, with minimal classification errors. Notice here that the axes are different than in linear regression: the x-axis is x_1 and the y-axis is x_2. To classify a point (x_1, x_2), you can check if it lies above or below the line.

As in regression, we add a component $x_0 = 1$ to each input $(x_1, \ldots, x_d)$ to model the intercept term. Notice that, without an intercept term, the linear classifier would be required to go through the origin, significantly skewing the solution. In Figure 11.2, for example, if $w_0 = 0$, then $w_1 x_1 + w_2 x_2 = 0$ for $(x_1 = 0, x_2 = 0)$. However, the line depicted clearly should not go through the origin.

The line is defined by the equation $\mathbf{x}^\top \mathbf{w} = 0$, for $\mathbf{x} = (1, x_1, x_2)$ and $\mathbf{w} = (w_0, w_1, w_2)$. Every vector (x_1, x_2) that satisfies the equation $\mathbf{x}^\top \mathbf{w} = 0$ is on this line. These vectors $\mathbf{x}$ are those that are orthogonal to $\mathbf{w}$, which means they have a 90 degree angle between them, as depicted in Figure 11.3. For example, if $\mathbf{w} = (0, 1, 1)$ (passes through the origin), then the vectors that are orthogonal are all of the form $(1, -t, t)$ for any $t \in \mathbb{R}$. This is because $\langle (1, -1, 1), (0, 1, 1) \rangle = 0 - 1 + 1 = 0$, and the constant t simply multiplies by zero. The constant t takes this orthogonal vector to $\mathbf{w}$ and then changes the magnitude, sliding it up and down this line, including flipping its sign. Notice that t is not multiplied by the intercept term, which always stays at 1; we only consider the changes to the inputs (x_1, x_2), since that is the only part of the input vector $\mathbf{x}$ that can change.

When shifting off of the origin, only w_0 changes, but (w_1, w_2) do not change. For example, again as seen in Figure 11.3, if we simply want to shift the line up, so that it goes through 0.5, then we set $w_0 = -0.5$. Now a new set of points (x_1, x_2) satisfy $\mathbf{x}^\top \mathbf{w} = 0$. Notice that the old choice of $(1, -1, 1)$ now is classified as a negative (below the line) because $\langle (1, -1, 1), (-0.5, 1, 1) \rangle = -0.5 - 1 + 1 = -0.5 < 0$. In other words, the new equation of the line is $(x_1, x_2)^\top (w_1, w_2) = -w_0 = 0.5$. For example, $\mathbf{x} = (1, 0, 0.5)$ are on this line, as is $\mathbf{x} = (1, 0.5, 0)$. This line is now defined by points $(0, 0.5) + t(1, -1)$ where the first term gives an offset from the origin. Now we have $(0, 0.5)^\top (w_1, w_2) + t(1, -1)^\top (w_1, w_2) + w_0 = 0.5 + t - t - 0.5 = 0$.

Exercise 29: Logistic regression learns a linear classifier on the given inputs. But, like

linear regression, this means we can get a version of polynomial logistic regression where we first transform the features using polynomials. What does the stochastic gradient descent updating look like for polynomial logistic regression? What might the resulting decision surface, separating the lines, look like? □

11.4 Issues with Minimizing the Squared Error

A natural question is why we went down this route for linear classification. Instead of explicitly assuming $p(y = 1|\mathbf{x})$ is a Bernoulli distribution and computing the maximum likelihood solution for $\sigma(\mathbf{x}^\top \mathbf{w}) = p(y = 1|\mathbf{x}, \mathbf{w})$, we could have simply decided to use $\sigma(\mathbf{x}^\top \mathbf{w})$ to predict targets $y \in \{0, 1\}$ and then tried to minimize their difference, using our favourite loss (the squared loss).

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \left(\sigma(\mathbf{x}_i^\top \mathbf{w}) - y_i\right)^2$$

Unfortunately, this more haphazard problem specification results in a non-convex optimization. In fact, there is a result that using the Euclidean error for the sigmoid transfer gives exponentially many local minima in the number of features [1].

Minimization of this non-convex function, therefore, is more problematic than the convex cross-entropy. Gradient descent on the cross-entropy, with stepsizes gradually decayed, will converge to a global minimum. For this non-convex squared error between the sigmoid prediction and the true label, it will generally be impossible to ensure we can converge to the global minimum. We could try to develop some tricks, like randomly restarting the optimization at different points to find different local minima, in the hopes that one will be the global minimum. But, such a search is not very effective. This example provides some motivation for why we care about selecting our objectives in an intelligent way.

We could also have stuck with squared errors, if we have used linear regression to learn a linear classifier. Now that you see logistic regression learns a linear classifier, you can see that we may have been able to learn this line with linear regression. If we set targets to be $y = 1$ for the positive class and $y = -1$ for the negative class, and learned $\mathbf{x}^\top \mathbf{w}$ to approximate these targets, we could obtain a solution where $\mathbf{x}^\top \mathbf{w} > 0$ means that we predict the positive class and $\mathbf{x}^\top \mathbf{w} < 0$ means we predict the negative class. In fact, this approach can work quite well. But, because it is attempting to actually predict targets y in a linear way, it can be skewed by high magnitude $\mathbf{x}$, whereas logistic regression is designed only to ensure the line separates points.

Consider the following example. We might have $\mathbf{x}_1 = (1, 1, 1)$ have $y_1 = 1$, $\mathbf{x}_2 = (1, 10, 10)$ have $y_2 = 1$ and $\mathbf{x}_3 = (1, -1, -1)$ have $y_3 = -1$. For logistic regression, this is no problem: the weights can be set to $\mathbf{w} = (0, 1, 1)$ to get $\mathbf{x}_1^\top \mathbf{w} = 2$, $\mathbf{x}_2^\top \mathbf{w} = 20$ and $\mathbf{x}_3^\top \mathbf{w} = -2$. When applying the sigmoid, these produce reasonable probability values $\sigma(\mathbf{x}_1^\top \mathbf{w}) = 0.88$, $\mathbf{x}_2^\top \mathbf{w} = 0.999$ and $\mathbf{x}_3^\top \mathbf{w} = 0.12$. For linear regression, however, it is hard to pick a $\mathbf{w}$ that predicts 1 for both $\mathbf{x}_1$ and $\mathbf{x}_2$. For a $\mathbf{w} = (0, 0.5, 0.5)$, we get a perfect prediction for $\mathbf{x}_1$ and $\mathbf{x}_3$, but $\mathbf{x}_2$ has a whopping squared error of $10^2 = 100$. To avoid getting a huge squared error for $\mathbf{x}_2$, it will learn a skewed $\mathbf{w}$.

Chapter 12

Bayesian Linear Regression

Bayesian estimation involves maintaining the entire posterior distribution, $p(\mathbf{w}|\mathcal{D})$. So far, we only looked at Bayesian estimation for a marginal distribution, such as for the case where $p(x)$ is a Poisson distribution with unknown parameter w corresponding to λ for the Poisson. We have only discussed how MLE and MAP extends to conditional distributions, $p(y|\mathbf{x})$. In this chapter, we make the next step: discussing Bayesian estimation for conditional distributions.

In fact, once we have looked at MAP, the extension to Bayesian estimation is not a big leap. For MAP, we already had to specify a prior to obtain $\text{argmax}_{\mathbf{w}\in\mathcal{F}}\, p(\mathbf{w}|\mathcal{D})$. For Bayesian estimation, we need to maintain the entire posterior $p(\mathbf{w}|\mathcal{D})$, not just the mode. Just as before, we simplify the explanation by only considering the univariate case: $w \in \mathbb{R}$.

12.1 The Posterior Distribution for a Known Noise Variance

Assume that $p(y|x) = \mathcal{N}(\mu = xw, \sigma^2)$ for some fixed $\sigma \in \mathbb{R}$. This is the assumption we made for linear regression, and then for MAP with a Gaussian prior on the weights. Again, let's assume a Gaussian prior on the weights $p(w) = \mathcal{N}(0, \sigma^2/\lambda)$ for some (regularization) parameter $\lambda > 0$. Then we get

$$\begin{aligned} p(w|\mathcal{D}) &= \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \\ &= \frac{p(w)\prod_{i=1}^n p(y_i|x_i,w)}{\int p(w)\prod_{i=1}^n p(y_i|x_i,w)dw} \end{aligned}$$

As in Section 5.3, computing the posterior is complicated by the integral in the denominator. In some cases, though, this integral can be solved analytically, and the posterior has a simple known form. This was the case with *conjugate priors*. For a given $p(y|x,w)$, a conjugate prior $p(w)$ is one where the posterior $p(w|\mathcal{D})$ is of the same form as the prior (example, both Gaussian).

For Bayesian linear regression, where $p(y|x) = \mathcal{N}(\mu = xw, \sigma^2)$, the conjugate prior is in fact the prior used for ℓ_2 regularization: $p(w) = \mathcal{N}(0, \sigma^2/\lambda)$. Given this prior, with prior mean $\mu_0 = 0$ and prior variance $\sigma_0^2 = \sigma^2/\lambda$, it can be derived that

$$p(w|\mathcal{D}) = \mathcal{N}(\mu_n, \sigma_n^2) \qquad \text{where} \qquad \sigma_n^2 = \frac{\sigma^2}{\sum_{i=1}^n x_i^2 + \lambda}$$

$$\mu_n = \frac{\sum_{i=1}^n x_i y_i + \lambda\mu_0}{\sum_{i=1}^n x_i^2 + \lambda} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda} = \frac{\sigma_n^2}{\sigma}\sum_{i=1}^n x_i y_i$$

The MAP solution corresponds to the mode of this distribution: μ_n.

Additionally, we can obtain a credible interval around weights that are plausible given the data, based on the variance σ_n. If the variance is big, then even after seeing the data there are many plausible values for w. As n gets larger, notice that σ_n^2 shrinks. We can be more precise by computing $p(w \in [a, b]|\mathcal{D}) = 0.95$ to get a 95% credible interval for w. Because the posterior is a Gaussian, we know just how to specify a and b to cover 95% of the density: $a = \mu_n - 1.96\sigma_n^2$ and $b = \mu_n + 1.96\sigma_n^2$.

You may be wondering why this looks so similar to a confidence interval, though it is called a credible interval. The reason is that both are built by reasoning about the distribution over our estimator, and integrating to get a probability of 95%. The key difference is that the distribution itself is different, because Bayesian approaches use a prior, as we show in the following example.

Example 26: In this example we show when a confidence interval and credible interval are similar. The difference arises because the credible interval is built by reasoning about the posterior, with a prior that restricted plausible outcomes. The confidence interval, on the other hand, does not incorporate such priors, and listens only to the observed data. They look similar when the prior for the credible interval is (nearly) uniform.

To understand why, let us return to the sample average estimator. Assume p_{true} is $\mathcal{N}(w_{\text{true}}, \sigma^2 = 1)$ with unknown mean w_{true}. Your goal is to estimate w_{true}. The sample average estimator—which actually corresponds to the maximum likelihood estimator—is $w(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n X_i$ which has variance $\sigma^2/n = 1/n$. Assuming a Gaussian distribution, we obtained 95% confidence interval $\Pr(w_{\text{true}} \in [w(\mathcal{D}) - 1.96/\sqrt{n}, w(\mathcal{D}) + 1.96/\sqrt{n}]) = 0.95$.

For the Bayesian approach, we assumed a Gaussian prior on w, $p(w) = \mathcal{N}(\mu_0, \sigma_0^2)$ and obtained posterior $p(w|\mathcal{D}) = \mathcal{N}(\mu_n, \sigma_n^2)$ where

$$\mu_n = \frac{1}{n + \sigma_0^{-2}} \left(\sum_{i=1}^n x_i + \frac{\mu_0}{\sigma_0^2} \right)$$
$$\sigma_n^2 = \frac{1}{n + \sigma_0^{-2}}$$

The resulting credible interval is $\Pr(w \in [\mu_n - 2\sigma_n, \mu_n + 2\sigma_n]) \geq 0.95$. If $\mu_0 = 0$ and σ_0 is very big—almost like having a uniform distribution—then $\mu_n \approx \frac{1}{n}\sum_{i=1}^n x_i$ and $\sigma_n^2 \approx 1/n$. The resulting credible interval is nearly the same as the confidence interval. Otherwise, the prior has an effect, primarily resulting in a tighter credible interval. □

12.2 The Posterior Distribution for Unknown Noise Variance

For linear regression, though, we typically do not know the variance σ^2. Fortunately, even when extending more generally to this setting, we have a conjugate prior. First consider the univariate case. We need now a prior on weights $w \in \mathbb{R}$ and also the variance σ^2. The conjugate prior is called the Normal-Inverse-Gamma (NIG) distribution, which has four parameters: $\mu_n, \lambda_n, a_n, b_n$. For prior parameters $\mu_0 \in \mathbb{R}$ and $\lambda_0, a_0, b_0 > 0$ (e.g.,

$\mu_0 = 0, \lambda_0 = 0.1, a_0 = 3, b_0 = 10$), we get posterior

$$p(w, \sigma^2|\mathcal{D}) = \text{NIG}(\mu_n, \lambda_n, a_n, b_n) \qquad \text{where} \quad \lambda_n = \sum_{i=1}^n x_i^2 + \lambda_0$$

$$\mu_n = \frac{\sum_{i=1}^n x_i y_i + \lambda_0 \mu_0}{\sum_{i=1}^n x_i^2 + \lambda_0} = \frac{\sum_{i=1}^n x_i y_i + \lambda_0 \mu_0}{\lambda_n}$$

$$a_n = a_0 + \tfrac{1}{2} n$$

$$b_n = b_0 + \tfrac{1}{2}\left(\sum_{i=1}^n y_i^2 + \lambda_0 \mu_0^2 - \lambda_n \mu_n^2\right)$$

For the NIG, the mode of the distribution is $\mathbb{E}[(w, \sigma^2)] = (\mu_n, \frac{b_n}{a_n - 1})$. The solution for w is the same as for MAP above. And now we also have an estimate for the most likely value for the variance of the noise $\frac{b_n}{a_n - 1}$.

This solution for the variance is not that intuitive; but for certain settings it is more intuitive. Let's assume that the data is centered, meaning $\frac{1}{n}\sum_{i=1}^n x_i = 0$. Let's further assume a simple choice on the prior parameters, $a_0 = 1, b_0 = 0$ and $\mu_0 = 0$, and use $\lambda = 0$ to minimally restrict the solution for w, giving $\lambda_0 = 0$. For this setting, $\mu_n = w_{\text{MLE}} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$. Let $\hat{y}_i = w_{\text{MLE}} x_i$. Then we get that

$$\begin{aligned}
\frac{b_n}{a_n - 1} &= \frac{b_0 + \frac{1}{2}\left(\sum_{i=1}^n y_i^2 + \lambda_0 \mu_0^2 - \lambda_n \mu_n^2\right)}{a_0 + \frac{1}{2} n} \\
&= \frac{\frac{1}{2}\left(\sum_{i=1}^n y_i^2 - \lambda_n \mu_n^2\right)}{\frac{1}{2} n} && \triangleright\ \mu_n = w_{\text{MLE}} \\
&= \frac{\sum_{i=1}^n y_i^2 - w_{\text{MLE}} \lambda_n \mu_n}{n} && \triangleright\ \lambda_n \mu_n = \lambda_n \frac{\sum_{i=1}^n x_i y_i + \lambda_0 \mu_0}{\lambda_n} = \sum_{i=1}^n x_i y_i \\
&= \frac{1}{n}\left(\sum_{i=1}^n y_i^2 - w_{\text{MLE}} \sum_{i=1}^n x_i y_i\right) \\
&= \frac{1}{n}\sum_{i=1}^n (y_i^2 - w_{\text{MLE}} x_i y_i) \\
&= \frac{1}{n}\sum_{i=1}^n (y_i^2 - \hat{y}_i y_i) \\
&= \frac{1}{n}\sum_{i=1}^n (y_i^2 - 2\hat{y}_i y_i + \hat{y}_i^2) - \frac{1}{n}\sum_{i=1}^n (\hat{y}_i^2 - \hat{y}_i y_i) \\
&= \frac{1}{n}\sum_{i=1}^n (y_i - \hat{y}_i)^2 - \frac{1}{n}\sum_{i=1}^n (\hat{y}_i^2 - \hat{y}_i y_i)
\end{aligned}$$

The first term $\frac{1}{n}\sum_{i=1}^n (y_i - \hat{y}_i)^2$ is a sample average estimate of the variance σ, using $\hat{y}_i$ as an estimate of $\mathbb{E}[Y|x_i]$. The second term reflects the additional variance due to the observed features themselves and the covariance between X and Y. The term $\frac{1}{n}\sum_{i=1}^n \hat{y}_i^2$ is a sample average estimate of the variance of the prediction, because an estimate of the variance is

given by $\frac{1}{n}\sum_{i=1}^n \hat{y}_i^2 - (\frac{1}{n}\sum_{i=1}^n \hat{y}_i)^2$ and

$$\frac{1}{n}\sum_{i=1}^n \hat{y}_i = \frac{1}{n} w_{\text{MLE}} \sum_{i=1}^n x_i = w_{\text{MLE}} \frac{1}{n}\sum_{i=1}^n x_i = 0$$

because the data is centered, i.e., $\frac{1}{n}\sum_{i=1}^n x_i = 0$. This variance is subtracted from the noise variance estimate, because we want to remove the influence of the feature variance on our estimate of the noise variance. The second term $\frac{1}{n}\sum_{i=1}^n -\hat{y}_i y_i$ is a sample covariance between the prediction and the observed target. This covariance needs to be added to the above, to account for the fact that $\hat{y}_i$ is not an independent estimate of $\mathbb{E}[Y|x_i]$—it is correlated with y_i because it uses that data to get the estimate.

We can again use this distribution to reason about a plausible set of values for the weights, namely the credible interval. The variance of the weights, under the NIG, corresponds to $\frac{b_n}{(a_n-1)\lambda_n}$ for $a_n > 1$. If this term is large, then the set of plausible weights are large. We can be more precise by computing $p(w \in [a,b]|\mathcal{D}) = 0.95$ to get a 95% credible interval for w. We can compute the marginal for w, of the NIG: it is a Student's t-distribution, with mean μ_n, scale parameter $\frac{a_n}{b_n\lambda_n}$ and degrees of freedom $2a_n$. Consequently, we can get a 95% credible interval using $[\mu_n - \epsilon, \mu_n + \epsilon]$ for $\epsilon = t_{0.05,2a_n}\frac{a_n}{b_n\lambda_n}$.

12.3 The Posterior Predictive Distribution

Practically, it is more useful to reason about the variability across our predictions using w, than about the variability in w itself. When we make a prediction $f(x)$, we would like to know the range of plausible values for that prediction. Namely, we would like to have $p(xw|x,\mathcal{D})$. We may additionally want to update our estimate of the conditional distribution, to get the *posterior predictive distribution*:

$$p(y|x,\mathcal{D}) = \int_w p(w|\mathcal{D})p(y|x,w)dw \tag{12.1}$$

This equation is effectively averaging over the predictions given by all w, proportionally to how likely each one is. It is a form of model averaging. Rather than picking one (likely incorrect) model, we can instead average over a set of possible models.

The elegance for Bayesian linear regression is that we can compute these distributions in closed form. First, we can compute the distribution over the prediction $p(xw|x,\mathcal{D})$, to get the credible interval for our prediction. The reason is simple: the random variable xw—random due to w—is still a Student's t-distribution, because w is a Student's t-distribution. Therefore, $p(xw|x,\mathcal{D})$ is a Student's t-distribution with mean $x\mu_n$, scale parameter $\frac{a_n x^2}{b_n\lambda_n}$ and degrees of freedom $2a_n$. Consequently, we can get a 95% credible interval using $[x\mu_n - \epsilon, x\mu_n + \epsilon]$ for $\epsilon = t_{0.05,2a_n}\frac{a_n x^2}{b_n\lambda_n}$. The form is even simpler under known variance σ^2, where $p(xw|x,\mathcal{D})$ is Gaussian with mean $x\mu_n$ and variance $x^2\sigma_n^2$.

We can use a similar approach to get the posterior predictive distribution. For a known variance σ^2, we have that $p(y|x,\mathcal{D}) = \mathcal{N}(x\mu_n, x^2\sigma_n^2 + \sigma^2)$. This involves the variance in our prediction, as well as the variance in y. For an unknown variance, $p(y|x,\mathcal{D})$ is a Student's t-distribution.

All the above extends to the multivariate case, the formulas simply become a bit more complex. We stayed in the simpler univariate setting, to avoid this complexity here. The

primary purpose of this chapter was to expose you to the ideas underlying Bayesian predictors, which can be quite different than the point estimates given by MLE and MAP. You should now be equipped with the tools to learn about the multivariate setting.

Chapter 13

Exercise Solutions

Exercise 5
This can be solved using the chain rule. Let $u(x) = -x^3$. Then we can rewrite $f(x)$ as $f(x) = \exp(u(x))$. We know that $\frac{du}{dx} = -3x^2$. Thus, using the chain rule, we get:

$$\frac{df}{dx} = \frac{df}{du}\frac{du}{dx} = \exp(-x^3)(-3x^2) = \boxed{-3x^2\exp(-x^3)}.$$

Exercise 6
Consider an event $A \in \mathcal{E}$. By property 1 of the event space, $A^c \in \mathcal{E}$. Since the intersection $A \cap A^c = \varnothing$, and we assume that P satisfies the axioms of probability, then by axiom 2 we have

$$\begin{aligned} P(A \cup A^c) &= P(A) + P(A^c) \\ P(\Omega) &= P(A) + P(A^c) \\ 1 &= P(A) + P(A^c) \\ P(A) &= 1 - P(A^c) \end{aligned}$$

Exercise 7
By property 3, we know $\mathcal{E} \neq \varnothing$. Thus there is some event $A \in \mathcal{E}$. By property 1, we know that $A^c \in \mathcal{E}$. By property 2 we then know that $A \cup A^c = \Omega \in \mathcal{E}$. By property 1, $\Omega^c = \varnothing \in \mathcal{E}$.
Exercise 8
The first axiom is straightforward to prove by combining the two statements we are given.

$$\begin{aligned} P(\mathcal{X}) &= \sum_{x \in \mathcal{X}} p(x) && \text{(by second statement)} \\ &= 1 && \text{(by first statement)} \end{aligned}$$

To prove the second axiom, first suppose that $A_1, A_2, \ldots \in \mathcal{E}$, $A_i \cap A_j = \varnothing \ \forall i, j$.

$$\begin{aligned} P(A_1, A_2, ... \in \mathcal{E}) &= \sum_{x \in \cup_i A_i} p(x) && \text{(by second statement)} \\ &= \sum_i \sum_{x \in A_i} p(x) && \text{(because } A_i \cap A_j = \varnothing\text{)} \\ &= \sum_i P(A_i) && \text{(be second statement)} \end{aligned}$$

Exercise 9
Using the power series expansion for the exponential, we have

$$e^\lambda = \sum_{x=0}^{\infty} \frac{\lambda^x}{x!}.$$

Thus,

$$\sum_{x=0}^{\infty} p(x) = \sum_{x=0}^{\infty} \frac{\lambda^x e^{-\lambda}}{x!} = \sum_{x=0}^{\infty} \frac{\lambda^x}{x!} e^{-\lambda} = \sum_{x=0}^{\infty} e^{\lambda} e^{-\lambda} = e^0 = 1.$$

Exercise 10

$$\begin{aligned}
\sum_y p(y|x) &= \sum_y \frac{p(x,y)}{p(x)} && \text{(definition of conditional probability)} \\
&= \frac{1}{p(x)} \sum_y p(x,y) \\
&= \frac{1}{p(x)} p(x) && \text{(marginalizing)} \\
&= 1
\end{aligned}$$

Exercise 11

$$\begin{aligned}
\mathbb{E}_{\mathcal{X}}[\mathbb{E}_{\mathcal{Y}}[Y|X]] &= \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} y p(y|x) \\
&= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} y p(y|x) p(x) \\
&= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} y p(x,y) \\
&= \sum_{y \in \mathcal{Y}} y \sum_{x \in \mathcal{X}} p(x,y) \\
&= \sum_{y \in \mathcal{Y}} y p(y) \\
&= \mathbb{E}[Y]
\end{aligned}$$

Exercise 12
Show that $\int_{\mathcal{X}} \left(\sum_{y \in \mathcal{Y}} f(x,y) p(x,y) \right) dx = \int_{\mathcal{X}} \mathbb{E}[f(x,Y)] p(x) dx$.

$$\begin{aligned}
\int_{\mathcal{X}} \left(\sum_{y \in \mathcal{Y}} f(x,y) p(x,y) \right) dx &= \int_{\mathcal{X}} \left(\sum_{y \in \mathcal{Y}} f(x,y) p(y|x) p(x) \right) dx \\
&= \int_{\mathcal{X}} \left(\sum_{y \in \mathcal{Y}} f(x,y) p(y|x) \right) p(x) dx \\
&= \int_{\mathcal{X}} \mathbb{E}[f(x,Y)] p(x) dx
\end{aligned}$$

Bibliography

[1] P Auer, M Herbster, and Manfred K Warmuth. Exponentially many local minima for single neurons. In *Advances in Neural Information Processing Systems*, 1996.

[2] A Banerjee, S Merugu, I S Dhillon, and J Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 2005.

[3] Peter L Bartlett and Shahar Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 2002.

[4] L on Bottou and Yann Le Cun. On-line learning for very large data sets. *Applied Stochastic Models in Business and Industry*, 2005.

[5] Léon Bottou. Online learning and stochastic approximations. *Online Learning and Neural Networks*, 1998.

[6] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to Statistical Learning Theory. In *Advanced Lectures on Machine Learning*. Springer Berlin Heidelberg, 2004.

[7] John C Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 2011.

[8] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer New York, 2013.

[9] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms - A Classification Perspective*. Cambridge University Press, 2011.

[10] Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the Complexity of Linear Prediction: Risk Bounds, Margin Bounds, and Regularization. In *Advances in Neural Information Processing Systems*, 2008.

[11] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

[12] Martha White. *Regularized factor models*. PhD thesis, University of Alberta, 2014.