Lecture Notes for

Machine Learning

An Introduction to Theory and Applications

Draft 2.2020|0.6

Gregory Wheeler
Professor of Philosophy and Computer Science
Centre for Human & Machine Intelligence
Frankfurt School of Finance & Management
g.wheeler@fs.de

Contents

Preface

"*As soon as it works, no one calls it AI any more.*" –John McCarthy

These lecture notes are designed to guide you through an introductory course in machine learning, and to serve as a bridge to the reference textbooks in the field, including (Bishop 2006; Hastie, Tibshirani, and Friedman 2009; Murphy 2012; Goodfellow, Bengio, and Courville 2016).

These notes are very much a work in progress, so your feedback is welcome. But because this is a working draft, I kindly ask that you not distribute these notes without my permission.

Gregory Wheeler
February 9, 2020, Frankfurt

Notation

Machine learning is an ensemble of methods and techniques from a variety of disciplines. Thus, there are many conventions for notation and terminology in use. Nevertheless, there is an emerging convention particular to Machine Learning, used for instance in (Goodfellow, Bengio, and Courville 2016), which attempts to strike a balance between familiarity, consistency, and convenience for particular purposes of Machine Learning. Nevertheless, overloading of notation is practically unavoidable, but context and explicit definitions in the text should suffice to resolve these ambiguities.

The following tables offer a guide to the notation conventions used in this book. Please consult the Appendix and Index for further information.

Numbers, Sets and Indexing Conventions

$\mathcal{X}$	A set
$x \in \mathcal{X}$	x is a member of $\mathcal{X}$
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integers
a	A scalar number; for our purposes either an integer or real number
$\boldsymbol{a}$	A vector
a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except the i-th element
$\boldsymbol{A}$	A matrix
$A_{i,j}$	Element (i, j) of the ith row and jth column of matrix $\boldsymbol{A}$
$\boldsymbol{a}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{a}_{:,j}$	Column j of matrix $\boldsymbol{A}$
$\mathbf{\mathsf{A}}$	A tensor
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-dimensional tensor A
$\{0, 1\}$	The set containing 0 and 1
$\{1, \ldots, n\}$	The set of integers between 1 and n.
$[a, b]$	The closed real interval including a and b
$[a, b)$	The closed-open (clopen) interval including a but excluding b

Functions

$f : \mathcal{X} \mapsto \mathcal{Y}$	The function f from domain $\mathcal{X}$ to range $\mathcal{Y}$
$f(\mathrm{x}; \boldsymbol{\theta})$	The function f of $x \in \mathcal{X}$ parameterized by $\boldsymbol{\theta}$; we sometimes omit $\boldsymbol{\theta}$ and write $f(\boldsymbol{x})$ instead
$\log x$	The natural logarithm of x.
$\ell(\boldsymbol{x}, y, f(\mathrm{x}))$	A loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \mapsto [0, \infty)$; we sometimes omit $\boldsymbol{x}$ and write $\ell(y, f(\mathrm{x}))$ instead
$J(\boldsymbol{\theta})$	A cost function
$\mathbf{1}_A$	The indicator function that assigns the value 1 if condition A is satisfied, 0 otherwise

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to vector $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Gradient of y with respect to matrix $\boldsymbol{X}$
$\nabla_{\mathsf{\boldsymbol{X}}} y$	Gradient of y with respect to tensor $\mathsf{\boldsymbol{X}}$
$\int_{\mathcal{A}} f(x)\, dx$	The definite integral over all $x \in \mathcal{A} \subseteq \mathcal{X}$

Probability and Statistics

Ω	Sample space
ω	A state in Ω
X	A random variable
$X \perp\!\!\!\perp Y$	The random variable X is independent of the random variable Y
$X \perp\!\!\!\perp Y \mid Z$	X is independent of Y conditional on Z
$p(A)$	The probability that the event A occurs
$p(x)$	The probability that the event $(X = x)$ occurs
$\mathbb{E}_p[X]$	The expectation of the random variable X with respect to probability p
$X \sim p$	Random variable X has distribution p
$H(X)$	Entropy of the random variable X
θ or $\boldsymbol{\theta}$	A parameter scalar or a parameter vector
$\hat{\theta}$	Estimate of a parameter
ϵ	Residual error
$h(\boldsymbol{x}; \boldsymbol{\theta})$	Estimator of y; also abbreviated as $h(\boldsymbol{x})$ or h

Datasets

$\mathcal{D}$	A finite set of m labeled or unlabeled training examples: Labeled if $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$ Unlabeled if $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{m}$
$\boldsymbol{x}^{(i)}$	The i-th training example (row vector) from $\mathcal{D}$
$\boldsymbol{x}_j$	The j-th feature (column vector) from $\mathcal{D}$
$x_j^{(i)}$	Value of the i-th training example (row) of the j-th feature (column) of the training set $\boldsymbol{X}$. Equivalently, the value of $\boldsymbol{X}_{i,j}$.
$y^{(i)}$	The target label associated with $\boldsymbol{x}^{(i)}$ for supervised learning

Miscellaneous

$\alpha := \beta$	α is defined by β (mathematical context)
$\alpha =: \beta$	β is defined by α (mathematical context)
$\alpha = \beta$	the assertion that α is equal to β (mathematical context)
$\alpha := \beta$	the operation of updating the current value assigned to variable α to the value of β (programming context)
$\alpha == \beta$	the assertion that α is equal to β (programming context)
$\boldsymbol{\theta}_{-\theta_i}$	The vector $\boldsymbol{\theta}$ whose elements remain fixed except for θ_i, which takes a value θ_i^* that is possibly different than θ_i; that is, for some θ_i^* that is not necessarily equal to θ_i, $\boldsymbol{\theta}_{-\theta_i} = [\theta_0, \theta_1, \ldots, \theta_{i-1}, \theta_i^*, \theta_{i+1}, \ldots]$,

Chapter 1

Introduction

What is machine learning?

> "*Field of study that gives computers the ability to learn without being explicitly programmed.*" – Arthur Samuel, 1959

1.1 What is a Learning Algorithm?

There is a particular notion of learning that machine learning algorithms are conceived to perform, which Tom Mitchell nicely characterizes in terms of a *task* to perform, the type of *experience* an algorithm is designed to receive, and some measure of *performance* that is used to evaluate how well the outcome of the algorithm succeeds in achieving the task:

> "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*" (Mitchell 1997, p. 2).

Common tasks that machine learning algorithms solve include *regression* and *classification* problems, where the goal is to predict a *scalar* value or *category* value, respectively. We will look at these two tasks and some others later.

Machine learning algorithms are commonly categorized by the type of experience they are designed to have of data during the learning process. **Supervised learning**, the most common of machine learning algorithms, allows the machine to experience *labeled data*. Labeled data is a data set where each training example consists of n features and a **label** or **target** value. For example, a housing training set might contain information about houses that sold in a city, where the target label for each home is its selling price along with features of each house, such as the size of its living area and number of rooms. **Unsupervised learning** algorithms experience unlabeled data, that is, a data set whose training examples have some number n of features but no labels. A common unsupervised learning problem is how to group together objects that are similar to one another into some number of *clusters*. In general, unsupervised algorithms are designed to discover structure within a data set. **Reinforcement learning** is a form of goal-directed, interactive learning that maps experience of an environment to actions that maximize a reward (Sutton and Barto 2018). Reinforcement learning algorithms are 'model-free', like unsupervised learning algorithms, but unlike unsupervised learning the aim of reinforcement learn is to maximize a reward signal rather than to find hidden structure. Reinforcement algorithms manage a trade-off between *exploring* new actions to take and *exploiting* the actions that are already known to yield a high reward, and is studied in a wide range of fields that concern boundedly rational decision-making (Wheeler 2018).

To evaluate how well a machine learning algorithm performs a given task we must define a quantitative measure of **performance**. Performance is understood in terms of an **objective function**, where the goal can be to minimize the number of errors, minimize the expected loss (as measured by the objective function), maximize the reward signal, et cetera. Often the goal is for a system to perform well on data that it has not seen before, which calls for evaluating the performance of a **test data set** that is distinct from the **training data set** used to train the model. Here there is another trade-off between **overfitting** a learning system on training data, which results in a learning system that achieves a very small loss on training data at the expense of comparatively high loss on the test data. Minimizing loss on training set data is typically not a good guide to minimizing loss in the out of sample test data.

1.2 Machine Learning and Statistics

Machine learning and statistics are both concerned with how to extract information from data. Although intimately related, the two fields are different in some important ways.

Statistics, the science of data, is divided into three branches: theory, methods, and applications. *Theory* asks after the properties of the mathematical objects used in statistical methods and what properties are preserved, or lost, by various operations on those objects. Statistical *methods* are techniques used to perform statistical analysis of data, which cover data collection and experimental design, exploratory data analysis, and statistical modeling. Finally, while statistical methods are applied within a range of domains, the development of statistical *applications* in some fields have contributed to statistical theory, statistical methods, or both. Examples include astronomy, biostatistics, physics, and psychology, among others.[1]

Machine Learning is a branch of Artificial Intelligence that developed primarily in three subfields of AI: *computer vision* (Prince 2012), *natural language processing* (NLP) (Manning and Schütze 1999), and robotics (Murphy 2019). All of these subfields confront problems that involve large scale, high-dimensional data. Historically, the nature of these problems have made them appear less suited to rule-based approaches to AI (Russell and Norvig 2009, Ch. 1) that are prominent in other subfields of AI, particularly *planning* (Ghallab, Nau, and Traverso 2016) and *knowledge representation and reasoning* (Fagin, Halpern, Moses, and Vardi 2003).

Although machine learning can be viewed as an area of applied statistics, doing so risks

[1]For an excellent history of statistics before the 20th century, see (Stigler 1986).

overlooking some distinct features of machine learning that are not shared by other areas of applied statistics. First, owing to its roots in computer science, machine learning emphasizes the computational costs of performing an operation. Sometimes an algorithm that uses a series of computationally cheap but highly imperfect estimators is surprisingly better than an alternative method that uses a few superior estimators that are computationally expensive. This **accuracy-effort trade-off** is a central focus of numerical computation in general, and of machine learning methods in particular.

Another important difference between statistics and machine learning is that machine learning techniques have historically focused narrowly on making accurate **predictions**, which is only one of several different types of **statistical inference** problems. Statistical methods are used to conclude whether a point estimate from one estimator is better than another, whether a sample is relevantly representative of the population from which it is drawn, and whether to reject the conclusion that one variable is independent of another. None of these problems have figured prominently in Machine Learning, although recent advances in computational methods in statistical modeling are bringing a computer science perspective to Bayesian statistics (McElreath 2016). Although statistical methods are also used to make predictions from observed data about unseen examples, techniques for making accurate predictions neither exhausts nor replaces the full scope of inferential statistics.

Third, owing to this difference in focus, the aim of statistics is different than the aim of machine learning. Traditional inferential statistics seeks to understand the relationship between variables of a model, adhering to what Leo Breiman has called a "data modeling" (Breiman 2001) culture. By contrast, the "algorithmic culture" of machine learning has traditionally sacrificed the goal of understanding the relationship between input x and output y in favor of accurately predicting the value of y from observed inputs x.

To illustrate this difference in viewpoint, imagine there is a data generating process of the following form,

$$y = \theta_0 + \theta_1 x + \epsilon. \tag{1.1}$$

Equation 1.1 describes a linear relationship between the outcome y and a single feature x in terms of the parameters, θ_0 and θ_1, and noise, ϵ. Usually the data generating process is unknown, so an important first step for traditional statistical inference is to select a model that reasonably approximates the available data generated by that process. If Equation 1.1 were selected, then the next step is to *estimate* the parameters θ_0 and θ_1, which are often interpreted as giving the causal effects of x on values of y, and estimate the noise by $\hat{\epsilon}$. For ordinary least squares regression, which we discuss in Chapter 2, the performance measure is *minimizing squared errors* by definition.

Machine learning, by contrast, tends to focus on the accurate predictions of y given x. Although Chapters 2 and 4 cover traditional regression and classification techniques from statistics, we nevertheless approach these models from a machine learning perspective—which is to say, in a nutshell, that we will focus on accurate predictions and less so on interpretability of our models. This points to another trade-off, between **prediction** and **explainability**, which is a source of controversy, for instance, when AI is use for automated decision-making. Without diminishing the moral and political importance of the issues that this prediction-vs-explanation tradeoff raises, it is nevertheless a remarkable scientific achievement of contemporary machine learning that we can create powerful models with high predictive accuracy without an interpretable model. Indeed, appreciating this capability is arguably necessary to grapple with the important societal implications machine learning methods.

1.3 Machine Learning and Data Science

Statistics *is* the science of data. Is 'data science', then, simply another name for statistics? Not quite. Whereas machine learning is primarily concerned with prediction problems in statistics and has extended the range of our capabilities to make accurate predictions, data science may be viewed as integrating statistics and computing more generally, including the computational infrastructure behind analysis, how to wrangle and manage large heterogenous data sets, and how to marshal these resources for the purpose of solving a concrete problem.

A word of caution about terminology. A parlor game of sorts can be played with defining the boundaries of statistics, data science, machine learning, and artificial intelligence, and specifying how each field is related to the others. The outline I've given above is not intended to be the final word but instead is offered as a first step to orient you to the theoretical and conceptual relationships among these rich and dynamic fields, and how the study of one might help you in the study of others.

Lastly, a word about explanations. Richard Feynman, in his famous illustration of *how explanations work*, pointed out that you need to take some things for granted to supply the terms to explain something else. Developing an ability to look at problems from both a computer science point of view and a statistical modeling point of view increases the number of things you can take for granted in order to explain a wider range of phenomena.

1.4 The Aim of this Book

What this book sets out to do is to introduce you to a computer science perspective on a range of statistical inference problems, and to do so in a manner that will help you move more fluidly between this perspective and a statistical point of view, when available, and to introduce to you some methods for constructing models. The terms I use therefore are selected to help you access large bodies of literature, and to begin to see the many ways these fields are related to one another.

Chapter 2

Regression Models

"Those who care to brace themselves to a sustained effort need not feel much regret that the road to be travelled over is indirect, and does not admit of being mapped beforehand in a way they can clearly understand. It is full of interest of its own. It familiarizes us with the measurement of variability, and with curious laws of chance that apply to a vast diversity of social subjects."

–Francis Galton, 1889

Suppose you have data about houses that sold in the Frankfurt area. Specifically, your data set includes the living areas x, in meters squared, and prices y, expressed in thousands of Euros, for m residential listings in Frankfurt.

	Size x in meters sq	**Price y in 1k Euros**
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
$\vdots$	$\vdots$	$\vdots$
m)	241.03	1091.10

Now you are asked, *"what is the relationship between living area and selling price?"* With this data set, how might you answer?

Regression models estimate the relationship between a target variable (e.g., price) and one or more feature variables (e.g., size). From a statistical point of view, specifying a model requires you to choose a **systematic** component and an **error** component of the model. Choosing a systematic component involves assessing the central tendency or "average" of the target variable relative to changes in magnitude of the independent feature variables, and the role of the error component is to specify what remains to be explained about the relationship between the target variable and the features by a model after it is fit to the data.[1]

In this chapter we will take a machine learning perspective on regression in general, and on linear regression in particular. Specifically, we will view regression as a supervised learning problem.

[1]A good textbook treatment of applied regression modeling is (Kleinbaum, Kupper, Nizam, and Rosenberg 2014).

2.1 Simple Linear Regression

A **regression function** summarizes the relationship between house size X and house price Y in Frankfurt in terms of the expected price of a house given its size:

$$h(x) = \mathbb{E}\left[Y|X=x\right] = \int y\, f(y|x)\, dy \tag{2.1}$$

The problem is that the regression function $h(x)$ is unknown. The aim of a **regression model** is to estimate $h(x)$ from the m observations of house-size and house-price pairs.

In the Frankfurt housing example, the living area of a particular house i, denoted by $x^{(i)}$, is the **feature** we will use to predict the price of house i, $y^{(i)}$, which is the **target** variable. For instance, the second house in the table has 82.67 square meters of living space and sold for 384.500 Euros. So, $x^{(2)} = 82.67$ and $y^{(2)} = 384.50$.

We say that a single **training example** or **observation** is the pair $(x^{(i)}, y^{(i)})$, and a finite set $\mathcal{D}$ of m training examples

$$\mathcal{D} = \left\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\right\}$$

is called a **training set** or a **data set**. Superscripts enclosed in parentheses is a convention for indexing the training set and should not be confused with the mathematical operation of exponentiation. For example, $x^{(2)}$ refers to the feature values of the second training example, whereas x^2 is the square of x. Figure 2.1 plots all m training examples in this data set.

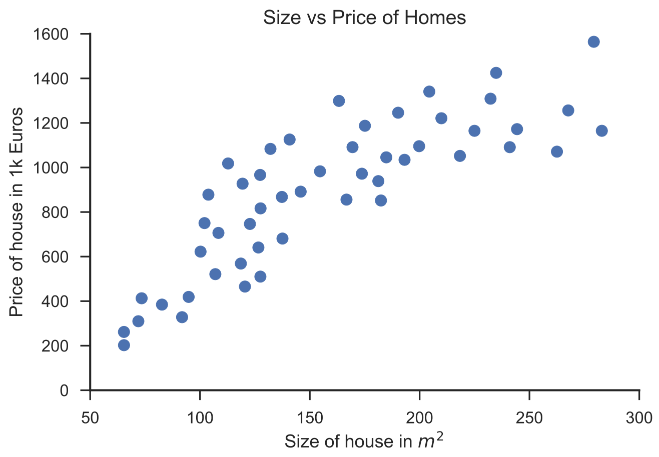


Figure 2.1: Frankfurt residential listings data set by size x (in meters sq) and price y (in thousands of Euros).

The simplest regression model is when there is a single feature, such as house size, and the regression function is assumed to be linear, that is

$$h(x) = \theta_0 + \theta_1 x \tag{2.2}$$

The difference between the actual sale price $y^{(i)}$ of a house and the price predicted by the regression function $h(x^{(i)})$ is an **error**, denoted by the error term $\epsilon^{(i)}$:

$$\epsilon^{(i)} = (y^{(i)} - h(x^{(i)})) \tag{2.3}$$

We assume that errors are unbiased in sign, positive or negative, that is that the expected value of an error is zero: $\mathbb{E}(\epsilon^{(i)}|X = x^{(i)}) = 0$. Otherwise, θ_0 can be adjusted to correct for a positive or negative bias. If we assume further that the variance of errors does not depend on the particular size of a house, that is $\mathrm{Var}(\epsilon^{(i)}|X = x^{(i)}) = \sigma^2$, then we have a **simple linear regression model**:

$$y = h(x) + \epsilon \tag{2.4}$$

Equation 2.4 specifies a linear systematic component $h(x)$ and an error component ϵ that is uncorrelated with observations.

Given the assumption that errors are unbiased, we may express our simple linear regression model in terms of the conditional expected value of y given x:

$$\mathbb{E}[y^{(i)}|x^{(i)}] = \theta_0 + \theta_1 x^{(i)} \tag{2.5}$$

Now we have the conditions under which Equation 2.5 is presumed to model Equation 2.1.

2.2 Ordinary Least Squares and Maximum Likelihood

Let's review. We started with an unknown regression function, Equation 2.1. By assuming that the regression function h is linear and that the errors ϵ are both unbiased and independent of the particular observed values of x, we transform the problem of confronting an unknown regression function to a problem of estimating the unknown parameters θ_0 and θ_1 and error term ϵ in Equation 2.4. Those parameters are estimated by the **ordinary least squares** method, which is the straight line plotted in Figure 2.2.

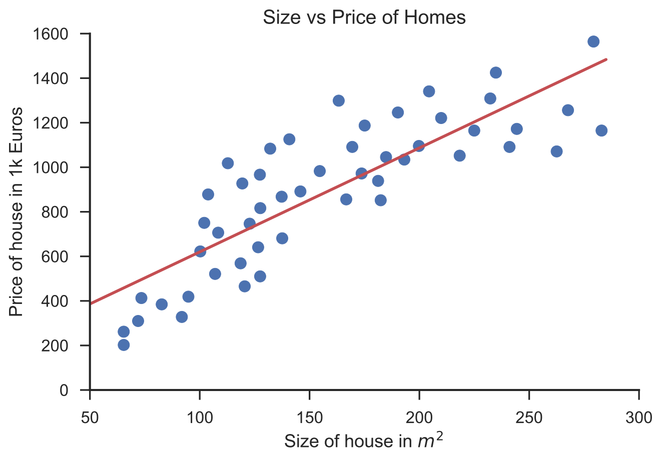


Figure 2.2: Frankfurt residential listings dataset. The line is the ordinary least squares line.

Set aside for a moment whether it is reasonable to assume that $h(x)$ is linear. (It isn't.) Turn instead to the method of fitting this simple linear model to our Frankfurt data set by ordinary least squares.

The error term ϵ in the simple linear regression model (Equation 2.4) is unknown, since both θ_0 and θ_1 are unknown. Let $\hat{\theta}_0$ and $\hat{\theta}_0$ denote **estimates** of θ_0 and θ_1. A **fitted**

line to data $\mathcal{D}$, $\hat{h}(x)$, is the line described by:

$$\hat{h}(x) = \hat{\theta}_0 + \hat{\theta}_1 x \tag{2.6}$$

and **fitted values** are $\hat{y}^{(i)} = \hat{h}(x^{(i)})$. The estimated values $\hat{\theta}_0$ and $\hat{\theta}_0$ are also called **regression coefficients**.

We can now estimate error for any pair of regression coefficients. A **residual** $\hat{\epsilon}^{(i)}$ is the difference between the actual value $y^{(i)}$ and the fitted value $\hat{y}^{(i)}$, and the discrepancy between actual values and the predicted values $\hat{y}^{(i)}$ of the fitted model $\hat{h}(x)$ is measured by the **residual sum of squares**:

$$RSS = \sum_{i=1}^{m} (\hat{\epsilon}^{(i)})^2 = \sum_{i=1}^{m} (y^{(i)} - (\hat{\theta}_0 + \hat{\theta}_1 x^{(i)}))^2 \tag{2.7}$$

The **ordinary least squares** (OLS) method picks the estimates $\hat{\theta}_0$ and $\hat{\theta}_1$ that minimize RSS. Theorem 2.1 specifies how these estimates are computed.

Ordinary Least Squares Estimates

Theorem 2.1. Let $\overline{x} = (x^{(1)} + \cdots + x^{(m)})/m$ and $\overline{y} = (y^{(1)} + \cdots + y^{(m)})/m$ The values $\hat{\theta}_0$ and $\hat{\theta}_1$ minimize RSS if and only if:

$$\hat{\theta}_1 = \frac{\sum_{i=1}^{m} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{m} (x_i - \overline{x})^2}$$

and

$$\hat{\theta}_0 = \overline{y} - \hat{\theta}_0 \overline{x}$$

Now suppose we add the assumption that errors are distributed according to the **normal distribution**, that is $\epsilon^{(i)}|x^{(i)} \sim \mathcal{N}(0, \sigma^2)$. This assumption says that the true values of the target y given features x is distributed normally with a mean of μ and variance of σ^2:

$$y^{(i)}|x^{(i)} \sim \mathcal{N}(\mu^{(i)}\sigma^2) \tag{2.8}$$

where $\mu^{(i)} = \theta_0 + \theta_1 x^{(i)}$.

With this new assumption, an alternative method is available for deriving the optimal parameters $\hat{\boldsymbol{\theta}}$ called the **maximum likelihood estimation** (MLE) method. The idea is simple: the MLE chooses $\hat{\boldsymbol{\theta}}$ which maximizes the likelihood of the sample. The **likelihood function** of a parameter value $\boldsymbol{\theta}$ on data $x^{(i)}$, $L(\boldsymbol{\theta}|x^{(i)})$, is the probability density of that data with those particular parameters.

To draw attention to the importance of assuming Gaussian noise, we present the likelihood function in terms of both $\boldsymbol{\theta}$ and σ:

$$\begin{aligned} L(\theta, \sigma) &:= -n \log \sigma - \frac{1}{2\sigma^2} \sum_{i}^{m} \left(y^{(i)} - (\hat{\theta}_0 + \hat{\theta}_1 x^{(i)}) \right)^2 \\ &= -n \log \sigma - \frac{1}{2\sigma^2} RSS \end{aligned} \tag{2.9}$$

Thus, maximizing $L(\theta, \sigma)$ is equivalent to minimizing the RSS.

Maximum Likelihood Estimates

Theorem 2.2. Suppose that the residuals $\hat{\epsilon}^{(i)}$ is additive, Gaussian, have constant variance, and are independent of $x^{(i)}$. Then, $\hat{\epsilon}$ is a maximum likelihood estimate if and only if $\hat{\epsilon}$ is an ordinary least squares estimate.

There are two points to emphasize. If any of the preconditions of Theorem 2.2 are not satisfied, then MLE estimates and OLS estimates can differ. Second, MLE assumes that residuals are Gaussian while OLS does not. They otherwise share the same assumptions. Note that OLS does not assume any particular probability distribution.

In some textbooks, the linear regression model is presented with the stronger assumptions that noise is Gaussian. This is especially the case when the model is expressed in the framework of **Bayesian statistics**.[2]

2.3 Matrix Format

There is an alternative format for the simple regression function, Equation 2.2, written in terms of matrix algebra. Reviewing how this simple equation for univariate linear regression can be rewritten in matrix format is important for understanding later chapters. The first step is to represent the two regression parameters in Equation 2.2 as a (2×1) **parameter vector $\boldsymbol{\theta}$**:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \tag{2.10}$$

The m observations of house sizes may also be thought of as a vector of length m. However, for reasons that will be given in a moment, these m observations are instead represented by a $(m \times 2)$ matrix, $\boldsymbol{X}$, that consists of a column of m house sizes, as you would expect, and a column of 1's, which you might not have expected:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix} \tag{2.11}$$

Equation 2.11 is called a **design matrix**. The first column of every design matrices consist of a column of 1's, called the **intercept terms**, followed by subsequent columns of feature values in the regression. Simple linear regression models have one column of feature values; multiple regression models have two or more columns of feature values. The reason for the column of intercept terms is to ensure that the dimensions

[2]A ordinary least squares linear regression is equivalent, in a Bayesian setting, to having a uniform prior, a normal likelihood, and calculating the posterior mode. If one or more of these conditions are unreasonable to assume, then the tools of Bayesian analysis would be helpful.

of our data matches the dimensions of our parameters support element-wise matrix operations.

$$\boldsymbol{X\theta} = \begin{bmatrix} \theta_0 + \theta_1 x^{(1)} \\ \theta_0 + \theta_1 x^{(2)} \\ \vdots \\ \theta_0 + \theta_1 x^{(m)} \end{bmatrix} \tag{2.12}$$

Design matrices, by definition, ensure that $\boldsymbol{X}$ and $\boldsymbol{\theta}$ are comformable. In general, matrix A multiplied by matrix B, written AB, is defined just in case A is **conformable** to B, that is, when the number of columns in A is equal to the number of rows in B.

With the addition of the intercept terms, we may rewrite Equation 2.2 as

$$h(x) = \sum_{i=0}^{n} \theta_i \boldsymbol{x}_i = \boldsymbol{X\theta} \tag{2.13}$$

where $n = 0, 1$ in our univariate case, $\boldsymbol{x}_0$ is the $(m \times 1)$ column vector of intercept terms, and $\boldsymbol{x}_1$ is the $(m \times 1)$ column vector of observed sizes of houses.[3]

The ordinary least squares estimate of the parameter vector $\boldsymbol{\theta}$ is given by the **normal matrix equation**, Theorem 2.3.

Normal Matrix Equation

Theorem 2.3. If matrix $\boldsymbol{X}^T\boldsymbol{X}$ is a normal matrix, then the ordinary least squares estimates $\hat{\boldsymbol{\theta}}$ are computed by

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Theorem 2.3 is an example of a formula that does not define an algorithm. Intuitively, when we select a vector of coefficients $\hat{\boldsymbol{\theta}}$ that minimizes squared error, we are minimizing the "size" of the vector of residuals $\hat{\boldsymbol{\epsilon}}$. Equation 2.7 defines residual sum of squares in terms of the scalar value difference, $\hat{\epsilon}^{(i)}$, between the scalar target $y^{(i)}$ and the scalar-valued prediction by $\hat{h}(x)$ for each observation i in the training set. The exact same quantities are minimized by Theorem 2.3, but are performed by operations on the vector $\hat{\boldsymbol{\epsilon}}$. First, observe that $\sum(\hat{\boldsymbol{\epsilon}}^{(i)})^2 = \hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$, which is the inner product of $\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}$:

$$\sum_{i=1}^{m} (\hat{\epsilon}^{(i)})^2 = \begin{bmatrix} \hat{\epsilon}^{(1)}, \hat{\epsilon}^{(2)}, \cdots, \hat{\epsilon}^{(m)} \end{bmatrix} \begin{bmatrix} \hat{\epsilon}^{(1)} \\ \hat{\epsilon}^{(2)} \\ \cdots \\ \hat{\epsilon}^{(m)} \end{bmatrix} = \hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}} \tag{2.14}$$

To show that Theorem 2.3 determines the values of the parameter vector $\boldsymbol{\theta}$ that mini-

[3]For a review of linear algebra, see Appendix C.

mizes $\left(\hat{\boldsymbol{\epsilon}}^T\boldsymbol{\epsilon}\right)$, the first step is to group terms.

$$\begin{aligned}
\min\left(\hat{\boldsymbol{\epsilon}}^T\boldsymbol{\epsilon}\right) &= \min_{\boldsymbol{\theta}}\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\theta}}\right)^T\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\theta}}\right)\\
&= \min_{\boldsymbol{\theta}}\left(\boldsymbol{y}^T-\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\right)\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\theta}}\right)\\
&= \min_{\boldsymbol{\theta}}\left(\boldsymbol{y}^T\boldsymbol{y}-\boldsymbol{y}^T\boldsymbol{X}\hat{\boldsymbol{\theta}}-\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{y}+\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}}\right) \qquad (2.15)
\end{aligned}$$

Next, Equation 2.15 is minimized by setting the derivatives with respect to $\hat{\boldsymbol{\theta}}$ to a vector of 0s.

$$\begin{aligned}
&-\boldsymbol{y}^T\boldsymbol{X}-(\boldsymbol{X}^T\boldsymbol{y})^T+2\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}=0\\
&-\boldsymbol{y}^T\boldsymbol{X}-(\boldsymbol{y}^T\boldsymbol{X})+2\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}=0\\
&2\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}=2\boldsymbol{y}^T\boldsymbol{X}\\
&\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}=\boldsymbol{y}^T\boldsymbol{X}\\
&\left(\hat{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}\right)^T=(\boldsymbol{y}^T\boldsymbol{X})^T\\
&\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}}=\boldsymbol{X}^T\boldsymbol{y} \qquad (2.16)
\end{aligned}$$

The last step is to isolate $\hat{\boldsymbol{\theta}}$ in Equation 2.16. The identity matrix $\boldsymbol{I}$ is

$$\boldsymbol{I}=(\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X}).$$

Since $\boldsymbol{I}\hat{\boldsymbol{\theta}}=\hat{\boldsymbol{\theta}}$, multiply both sides of Equation 2.16 by $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ to isolate $\hat{\boldsymbol{\theta}}$:

$$\begin{aligned}
(\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}}) &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{y})\\
\hat{\boldsymbol{\theta}} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \qquad (2.17)
\end{aligned}$$

Equation 2.17 is the **normal equation** and $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$ is called the **Moore-Penrose pseudo-inverse**.

2.4 Polynomial and Multiple Regression

We might reasonably ask whether a linear regression model, Equation 2.2, is the appropriate regression model for this data set. Intuitively, there is a slight curve in the data that a linear hypothesis ignores, meaning that $h(x)$ will underestimate the price of medium-sized homes and overestimate the price of both small homes and large. Put directly, one of the assumptions of the simple linear model is that the variance of errors does not depend on the particular size of the house, but this assumption is not satisfied with respect to the data set $\mathcal{D}$.

Instead of a linear hypothesis h, we might instead propose a regression model that better captures this non-linearity in the data. One option is to replace the linear model in Equation 2.2 with a second-order polynomial h^*:

$$h^*(x)=\theta_0+\theta_1 x+\theta_2 x^2$$

Notice that h^* operates on the same input of data, but introduces a new feature that is a function of the input, x^2, along with a new adjustable parameter θ_2. So, instead of explaining house prices as a linear function of their size x, h^* proposes to explain house prices as a linear function of house size x and the square of house size, x^2. Figure 2.3 plots the ordinary least squares model fitting $h^*_\theta(x)$, in green, together with the original ordinary least squares model of h as a red dashed line.

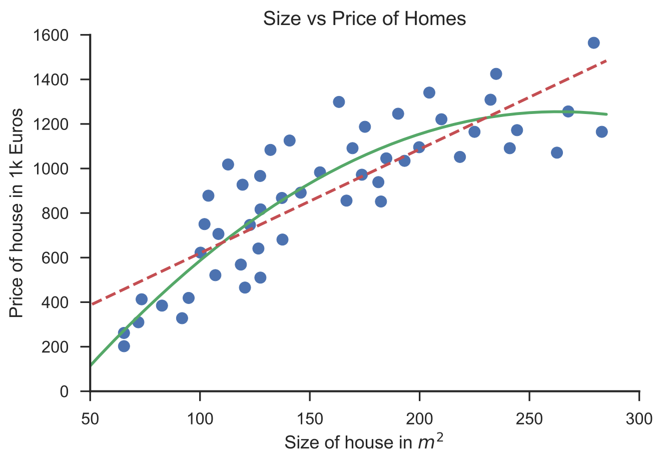


Figure 2.3: Frankfurt residential listings dataset fit by a simple linear model (dash) and a second-order polynomial regression model (solid) Both lines are fit by OLS.

Choosing between h and h^* is a **model selection** problem. We will have more to say about model selection problems in later chapters.

In addition to **polynomial regression** functions, which engineer new independent features from the existing input from the data set, we may also consider data sets that provide multiple independent features as input. For example, suppose that in addition to knowing the living space of homes sold in Frankfurt you also know the number of rooms, the number of parking places, and how long ago the house was built.

	Size in m^2	Rooms	Parking Places	Age in years	Price (1k)
	x_1	x_2	x_3	x_4	y
1)	52.14	1	0	95	164.21
2)	82.67	2	1	38	384.50
3)	120.47	3	1	41	465.41
⋮			⋮	⋮	
m)	241.03	7	3	1	1091.10

Multiple regression extends simple linear regression model by using two or more feature variables to predict a target scalar variable. Now the advantages of the matrix form of linear regression from Section 2.3 come into view. For notice that the $(m \times 5)$

design matrix $\boldsymbol{X}$ for our 4 features is conformable to the (5×1) parameter vector $\boldsymbol{\theta}$,

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & x_4^{(m)} \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \tag{2.18}$$

which allows us to apply Equation 2.13:

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \theta_0 + \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ \vdots \\ x_1^{(m)} \end{bmatrix} \theta_1 + \begin{bmatrix} x_2^{(1)} \\ x_2^{(2)} \\ \vdots \\ x_2^{(m)} \end{bmatrix} \theta_2 + \begin{bmatrix} x_3^{(1)} \\ x_3^{(2)} \\ \vdots \\ x_3^{(m)} \end{bmatrix} \theta_3 + \begin{bmatrix} x_4^{(1)} \\ x_4^{(2)} \\ \vdots \\ x_4^{(m)} \end{bmatrix} \theta_4 = \boldsymbol{X\theta} \tag{2.19}$$

Equation 2.19 makes clear that the matrix $\boldsymbol{X}$ multiplied by the vector $\boldsymbol{\theta}$ is the linear combination of the columns of $\boldsymbol{X}$ and the elements of $\boldsymbol{\theta}$, which act as their name implies as coefficients.

2.5 Simple Supervised Learning

Fitting a simple linear regression model is, from a machine learning perspective, a learning problem in which the objective is to find the optimal coefficient parameters $\hat{\boldsymbol{\theta}}$. Intuitively, a learning algorithm fits an OLS regression by taking a labeled data set $\mathcal{D}$ as *input* and using RSS as a numerical *performance measure* to evaluate candidate values for $\hat{\boldsymbol{\theta}}$. The parameter vector $\hat{\boldsymbol{\theta}}$ is optimal just in case there is no other parameter values $\hat{\boldsymbol{\theta}}^*$ that performs better than $\hat{\boldsymbol{\theta}}$. The purpose of a simple linear regression model then is to take an input of features $\boldsymbol{x}$ to predict the value of a scalar $y \in \mathbf{R}$. This focus on parameter optimization and a procedural, input-output relationship between features and targets motivates two alterations to our notation.

First, to make explicit the important role of the parameter vector $\boldsymbol{\theta}$, we can express the regression function in Equation 2.1 as a **parametric model**:

$$h(x; \boldsymbol{\theta}) = \mathbb{E}\left[Y | X = x; \boldsymbol{\theta}\right] = \int y\, f(y|x; \boldsymbol{\theta})\, dy \tag{2.20}$$

The conditional density $f(y|x; \boldsymbol{\theta})$ expresses the probability of y given the value x of the random variable X, where $\boldsymbol{\theta}$ is a finite set of parameters which specify all $f \in F$ densities of the model. For example, if we assume that data X are generated by a normal distribution with mean μ and variance σ^2, $x \sim \mathcal{N}(\mu, \sigma^2)$, then the set F of unconditional densities will consist of all densities of the form $f(x; \mu, \sigma)$. If $\boldsymbol{\theta}$ is a vector but we are only interested in one component, we refer to the remaining parameters in that context as **nuisance parameters**. For instance, if we are interested in estimating the central tendency of x but do not care about dispersion, then σ would be set aside as a nuisance parameter to focus on $f(x;\mu)$.

This description of OLS linear regression as a parametric model draws attention to the distinct task of learning $\hat{\boldsymbol{\theta}}$. The closed form OLS solution, Theorem 2.3, is neither an efficient nor comprehensive algorithm for computing optimal $\hat{\boldsymbol{\theta}}$. The asymptotic time complexity of computing closed-form normal matrix solution to ordinary least squares regression is exponential in the number of features: $\mathcal{O}(n^2m)$. Thus, for data sets with a large number of feature variables, exact solutions are computationally infeasible. In addition, Theorem 2.3 assumes that the number of features is less than the number of observations, $n \leq m$, otherwise the design matrix fails to be invertible. So, for data sets in which the number of features exceeds the number of observations, Theorem 2.3 is inapplicable.

Intuitively, the simple regression function h in Equation 2.2 is an hypothesis asserting how house size is functionally related to its selling price. Given that hypothesis, we would like to select values for θ_0 and θ_1 so that the predicted value $h(x^{(i)})$ is "close" to $y^{(i)}$ for each i in our training set. A **loss function** is any function that measures the discrepancy between predicted values by h and true values $y^{(i)}$ in a data set. RSS is one of several loss functions,

$$\ell\left(y^{(i)}, h(x^{(i)})\right) = \left(h(x^{(i)}) - y^{(i)}\right)^2, \text{ for each } i \tag{2.21}$$

We will introduce other loss functions in Chapter 5.

A loss function specifies how differences between *individual* predictions and true label values are evaluated. A **cost function** J, in turn, defines how these individual losses are used in an overall evaluation of the hypothesis h across many observations. For example, **mean squared error** (MSE), is $\frac{1}{m}RSS$ (for m observations):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - h(x^{(i)};\boldsymbol{\theta}))^2 = \frac{1}{m}\sum_{i=1}^{m}(\hat{\epsilon}^{(i)})^2 \tag{2.22}$$

Although from a mathematical point of view RSS and MSE work equally well for optimization, since $\frac{1}{m}$ is a constant, for numerical optimization algorithms a normalizing term such as $\frac{1}{m}$ is important. For this reason, we define the cost function $J(\boldsymbol{\theta})$ for linear regression defined by Equation 2.23:

$$J(\boldsymbol{\theta}) = \frac{1}{2}\cdot\frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)};\boldsymbol{\theta}\right) - y^{(i)}\right)^2 \tag{2.23}$$

The addition of the constant $\frac{1}{2}$ is for mathematical convenience, which will become apparent in Chapter 3. It also does not affect the optimization of $J(\boldsymbol{\theta})$ for the same reason: it is a constant.

Finally, the OLS regression coefficients $\hat{\boldsymbol{\theta}}$ are those which minimize $J(\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}} = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)};\boldsymbol{\theta}\right) - y^{(i)}\right)^2 \tag{2.24}$$

Hereafter we will refer to Equation 2.24 as the optimization objective for OLS linear regression models.

It should be noted that some machine learning textbooks, such as (Goodfellow, Bengio, and Courville 2016), use the terms ‘loss function’, ‘cost function’, and ‘error function’ interchangeably. This is flexibility in use is not uncommon. Consider the term ‘wheels’, which may be used to refer solely to the metal cylinders attached to the axels of the car, those plus the tires, or, colloquially, the whole vehicle. This varied use does not undermine the fact that these are different things, but rather points to circumstances in which these differences are safely ignored.

A second difference to emphasize is the input-output nature of learning problems, which is made to align notation closer to the form of the input data to a learning algorithm. Equation 2.19 makes plain that the matrix form of the normal equation operates column-wise. However, data is collected and typically input to a learning algorithm as row-wise observations i of $\mathbf{X}$. Moreover, we may wish to calculate an estimate of an individual training example i. Finally, although all vectors in a design matrix are of the same length, by definition, in practice input data to other algorithms can be of different lengths. For these reasons, it is common to represent individual training examples as an **input vector**. For example, each house i in the Frankfurt housing data set can be represented as 5-dimensional vector, $\boldsymbol{x}^{(i)} \in \mathbb{R}^5$. Specifically, $\boldsymbol{x}^{(i)} = [x_0^{(i)}, x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}]$ consists of $x_0^{(i)} = 1$, the intercept term; $x_1^{(i)}$ is the living area of house i (in units m^2); $x_2^{(i)}$ is the number of rooms; $x_3^{(i)}$ is the number of dedicated parking places; and $x_4^{(i)}$ is the current age of the house in years.

The important difference is that the input vector $\boldsymbol{x}^{(i)}$ and the parameter vector $\boldsymbol{\theta}$ are *both* conceived as column vectors of the same dimension. So, their linear combination can be computed by the inner product of $\boldsymbol{x}^{(i)}$ and $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \cdots \theta_i x_m^{(m)} \tag{2.25}$$

where, recall, $x_0^{(i)} = 1$. We sometimes write $\boldsymbol{\theta}^T \boldsymbol{x}$ to refer to the form of the all m observations.

Hereafter we largely stop decorating variables with hats to distinguish between estimated quantities – such as $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\epsilon}}, \hat{\boldsymbol{h}}(\boldsymbol{x}), \hat{\boldsymbol{y}}$– and their corresponding unknown population quantities, unless we encounter a context in which the distinction between estimates and true parameters is salient. This forgoing of hats strikes a balance between precision in notation and readability.

Chapter 3

Gradient Descent

Minimizing the loss function in equation 2.24 is a routine task, but most implementations of ordinary least squares regression hide how the **optimization problem** of choosing $\boldsymbol{\theta}$ so as to minimize $J(\boldsymbol{\theta})$ is performed. Moreover, nearly all machine learning software libraries hide these details too, and for good reason. Efficient numerical optimization algorithms are exceedingly difficult to implement. In practice, unless you are an expert in soft-computing and numerical optimization, you are much better off using built-in advanced optimization packages just as you are better off using the built-in square-root function than to build your own from scratch. However, important insights into the nature of this optimization problem and how a machine can efficiently yield numerical approximate solutions are gained by implementing an algorithm named **LMS** to compute **least mean squares**. The simple version of this algorithm that we will implement does not even work that well in practice. Yet, here again, understanding why it doesn't work so well, and why alternative methods have been developed, will also help you conceptually to understand how numerical optimization problems are solved by computational methods. The time invested to understand this algorithm will help you later when you turn to more complicated learning algorithms.

The basic idea is this. Imagine an n-dimensional space of possible input values to some function J, where n is the dimension of the parameter vector $\boldsymbol{\theta}$. For example, univariate ordinary least squares regression is the problem of finding values for $\boldsymbol{\theta} = [\theta_0, \theta_1]$ where the loss function J is defined by Equation 2.24 and the value of $J(\boldsymbol{\theta})$ is minimized. This is the minimization problem discussed in chapter 2 with respect to h. What the LMS algorithm does for this simple problem is start by picking arbitrary values for θ_0 and θ_1, which may be thought of as an initial guess at the parameter values that minimize J. The gradient of θ_0 and θ_1 is calculated with respect to the cost function $J(\boldsymbol{\theta})$, and changes to θ_0 and θ_1 are made in the direction of the negative gradient to reduce $J(\boldsymbol{\theta})$, if possible, and this procedure is repeated until, *hopefully*, we converge to values for θ_0 and θ_1 that minimizes $J(\boldsymbol{\theta})$.

A very powerful and general optimization algorithm called **gradient descent** can be used to implement this "first guess, then iteratively improve" approach.

3.1 Gradient Descent

You may recall from calculus that, for an input vector $\boldsymbol{\theta}$ in $\mathbb{R}^n$ to some function J, the gradient of $J(\boldsymbol{\theta})$ points in the direction of steepest increase in the value of J within this n-dimensional input space. For example, $\boldsymbol{\theta} = [\theta_0, \theta_1]$ is in $\mathbb{R}^2$, the 2-dimensional input to the loss function J for univariate linear regression, and the (non-zero) gradient of $J(\boldsymbol{\theta})$ would yield a vector $\boldsymbol{\theta}^* = [\theta_0^*, \theta_1^*]$ whose values “point the direction” in $\mathbb{R}^2$ to the steepest *increase* in the value of J. The gradient of $J(\boldsymbol{\theta})$ is zero if there is no $\boldsymbol{\theta}^*$ that increases J, in which case the vector $\boldsymbol{\theta}$ is such that $J(\boldsymbol{\theta})$ is a local maximum. Gradient descent aims to *minimize* J, so we will work with the **negative gradient** and seek to find a **local minimum**.

Gradient descent starts with an initial parameter vector $\boldsymbol{\theta}$ in $\mathbb{R}^n$ and then computes a new vector $\boldsymbol{\theta}^*$ to replace $\boldsymbol{\theta}$ in accordance with the **step direction** in $\mathbb{R}^n$, computed by the (negative) gradient $\nabla J(\boldsymbol{\theta})$, and the **step size** or **learning rate** α which controls the size of the changes to $\boldsymbol{\theta}$ at each iteration.[1]

To make this computation more concrete, focus on the ith component of the vector $\boldsymbol{\theta}$, written θ_i, and consider how to update scalar θ_i to a new scalar θ_i^* from the step direction and step size computed for the cost function $J(\boldsymbol{\theta})$:

$$\theta_i^* = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta}) \tag{3.1}$$

where $\frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta})$ is the partial derivative of the ith component of $\boldsymbol{\theta}$ with respect to the cost function J at iteration k, $0 < \alpha$ is the learning rate, θ_i is the value of the ith component of $\boldsymbol{\theta}$, and the minus sign denotes that we are using the negative gradient to update θ_i to θ_i^*. Intuitively, Equation 3.1 expresses the difference between the original value of θ_i and a step in a positive or negative direction in the i-dimension of $\boldsymbol{\theta}$, depending on whether the slope of $\frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta})$ is negative or positive, respectively.

Each element θ_i of $\boldsymbol{\theta}$ is **simultaneously updated**, meaning that the new value θ_i^* for each element i of $\boldsymbol{\theta}$ is changed with respect to the current values of the remaining elements, $\boldsymbol{\theta}_{-\theta_i}$:

$$\begin{aligned}
\text{update to } \theta_0 \text{ is wrt } \boldsymbol{\theta}_{-\theta_0} &= [\theta_0^*, \theta_1, \ldots, \theta_i, \ldots, \theta_n] \\
\text{update to } \theta_1 \text{ is wrt } \boldsymbol{\theta}_{-\theta_1} &= [\theta_0, \theta_1^*, \ldots, \theta_i, \ldots, \theta_n] \\
\vdots \qquad\qquad & \qquad\qquad \vdots \\
\text{update to } \theta_i \text{ is wrt } \boldsymbol{\theta}_{-\theta_i} &= [\theta_0, \theta_1, \ldots, \theta_i^*, \ldots, \theta_n] \\
\vdots \qquad\qquad & \qquad\qquad \vdots \\
\text{update to } \theta_n \text{ is wrt } \boldsymbol{\theta}_{-\theta_n} &= [\theta_0, \theta_1, \ldots, \theta_i, \ldots, \theta_n^*]
\end{aligned}$$

The algorithm repeats this update procedure iteratively until, hopefully, the updated values of $\boldsymbol{\theta}$ converge to a stable minima for $J(\boldsymbol{\theta})$, which occurs when the partial derivative terms for each element θ_i is zero, that is, when $\theta_i - (\alpha \cdot 0)$. Algorithm 3.1 presents the **gradient descent algorithm**.

[1]The terminology ‘step direction’ and ‘step size’ is standard but misleading. We say that $\nabla J(\boldsymbol{\theta})$ is the ‘step direction’, even though it does not necessarily have unit norm, and that α is the step size even though the difference between the kth iteration to the updated $k+1$st iteration is not equal to α unless the search direction has the unit norm, that is $\|\theta^{(k+1)} - \theta^{(k)}\|$ only if $\|\nabla J(\boldsymbol{\theta}^{(i)})\| = 1$.

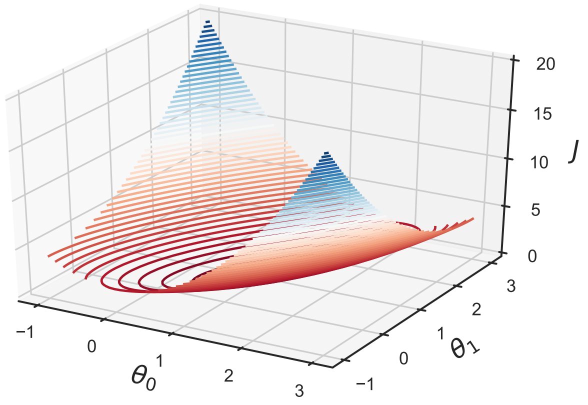


Figure 3.1: Surface plot of $J(\boldsymbol{\theta})$ in Equation 3.2.

Gradient Descent

Algorithm 3.1.

```
1: Initialize parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_n]$
2: repeat
3:     $\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta)$
4:     $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta)$
5:     $\dots$                                        ▷ Simultaneous update $\theta_i$'s
6:     $\theta_n := \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta)$
7: until Convergence
8: Return $\boldsymbol{\theta}$
```

Note in Algorithm 3.1 use the notation ':=' to denote that the value assigned to the variable on the lefthand side is updated by the value computed on the righthand side.

Gradient descent is a particular kind of steepest descent method, where intuitively the goal is to pick values in all components of a vector to minimize a cost function as quickly as possible. Consider an example. The vector $\boldsymbol{\theta} = [\theta_0, \theta_1]$ minimizes the equation

$$J(\boldsymbol{\theta}) = (\theta_0 - \theta_1)^2 + (\theta_0 - 1)^2 \tag{3.2}$$

when $\theta_0 = \theta_1 = 1$. The 3-dimensional mesh plot in Figure 3.1 visualizes the values of θ_0 and θ_1 that minimize $J(\boldsymbol{\theta})$, as defined by Equation 3.2, where z-axis represents $J(\boldsymbol{\theta})$, and the x- and y-axis represent θ_0 and θ_1, respectively.

Alternatively, we can represent the same information in a 2-dimensional contour plot in Figure 3.1. Included in this contour plot is a trace of values that we might pick to find the minimum of the function $J(\boldsymbol{\theta})$, starting with the values $\theta_0 = -1$ and $\theta_1 = 0$ and ending at $\theta_0 = \theta_1 = 1$. To apply this method to the ordinary least squares regression model from Chapter 2, we must derive the derivative terms to plug into the gradient descent algorithm from Equation 2.24.

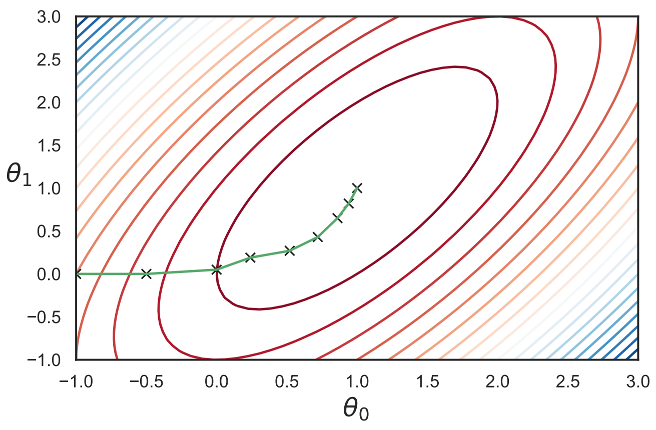


Figure 3.2: Contour plot of $J(\boldsymbol{\theta})$ in Equation 3.2

Suppose $\boldsymbol{\theta}$ is a parameter vector and θ_i is the ith element of $\boldsymbol{\theta}$. Then, we have

$$J(\boldsymbol{\theta}) = \frac{1}{2}\left(h\left(x\right) - y\right)^2$$

which, for each θ_i of $\boldsymbol{\theta}$, is

$$\begin{aligned}
\frac{\partial}{\partial\theta_i}J(\boldsymbol{\theta}) &= \frac{\partial}{\partial\theta_i}\frac{1}{2}\left(h\left(x\right) - y\right)^2 \\
&= \frac{1}{2}\left(\frac{\partial}{\partial\theta}\left(h(x) - y\right)^2\right) \\
&= \frac{1}{2}\cdot 2\left(h\left(x\right) - y\right)\cdot\frac{\partial}{\partial\theta_i}\left(h\left(x\right) - y\right) \\
&= \left(h\left(x\right) - y\right)\cdot\frac{\partial}{\partial\theta_i}\left(h\left(x\right) - y\right) \\
&= \left(h\left(x\right) - y\right)\cdot\frac{\partial}{\partial\theta_i}\left(\theta_0 + \theta_1 x - y\right) \\
&= \left(h\left(x\right) - y\right)x_i
\end{aligned}$$

where $x_0 = 1$, by convention. Thus, a single training example k will have the update rule of the form:

$$\theta_i := \theta_i - \alpha\left(h(x^{(k)}) - y^{(k)}\right)x_i^{(k)}$$

whereas updating θ_i by the **full batch** of all m training examples in a training set will update by the average:

$$\theta_i := \theta_i - \alpha\frac{1}{m}\sum_{j=1}^{m}(h_\theta(x^{(j)}) - y^{(j)})x^{(j)} \tag{3.3}$$

Finally, a remark on the learning rate α. When α is too small, gradient descent is very slow to converge. Yet if α is too large, gradient descent may not decrease, and may either jump to another local minima or fail to converge at all. Before considering how to choose a value for α, we might ask whether there are any guarantees that gradient descent will converge. The answer is yes, but they come with caveats. If we assume that our cost function J is convex and differentiable, then there is a theorem which says that there is some constant $\mathcal{L}$, called the *Lipschitz constant*, such that gradient descent will converge with a fixed learning rate α that is bounded by $\frac{1}{\mathcal{L}}$.

Convergence for Fixed Learning Rate

Theorem 3.1. Suppose f is a convex and differentiable function from $\mathbb{R}^n$ to $\mathbb{R}$, $\mathcal{L} > 0$ is a constant, and ∇f is **Lipschitz continuous**, that is

$$\|\nabla f(x) - \nabla f(y)\| \leq \alpha\|x - y\| \quad \text{for } x, y \in \mathbb{R}^n$$

Then, gradient descent with fixed learning rate $\alpha \leq \frac{1}{\mathcal{L}}$ converges.

What this theorem says is that you will converge if you choose $1/\mathcal{L}$ as your learning rate. So, however conservative you are, you should choose at least $1/\mathcal{L}$. For example, if you have a function f whose gradient changes very slowly, the Lipschitz constant c in this case is small, and the step size is large. The Lipschitz constant only gives you a lower bound on step size, however. In practice you should look for a larger step size even though there is no theorem to guarantee a larger step size will converge.

How useful is Theorem 3.1? This theorem gives a lower bound on the step size α when what we would like, ideally, is an upper bound. But even more worrying is that, in practice, we often do not know for sure that an objective function is convex and differentiable. Moreover, nearly all convergence theorems invoke conditions that are typically unknown. So, why bother?

The short answer is that convergence theorems such as Theorem 3.1 can nevertheless give you important conceptual insights into a learning method. While it is true that solving optimization problems is largely an engineering task, best practices learned empirically about a particular method are invariably guided by a theoretical understanding of the methods put to use.

3.2 General Gradient Methods

Gradient descent belongs to a family of **descent methods**, which are algorithms that produce a minimizing sequence of updates to a vector $\boldsymbol{\theta}$ in $\mathbb{R}^n$ in terms of a **search direction** $\Delta\boldsymbol{\theta}$, also in $\mathbb{R}^n$, and a scalar **step size** α (Cauchy 1847). Algorithm 3.1 is a particular type of **general descent method**. This slight generalization, described in Algorithm 3.2, will allow us to consider some variations of gradient descent. The reason we will consider variations of gradient descent is that Algorithm 3.1 does not work well in practice and it is not obvious why this should be the case. Yet understanding why this is so is the second main reason for studying this algorithm, for understanding why will help you when you confront less intuitive methods that are effective in practice.

General Descent Method

Algorithm 3.2.

```
1: Initialize parameters $\boldsymbol{\theta}$.
2: repeat
3:     Determine a descent direction $\Delta\boldsymbol{\theta}$.
4:     Perform line search by choosing a step size $\alpha > 0$.
5:     Update $\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}$.
6: until A stopping criterion is satisfied
7: Return $\boldsymbol{\theta}$
```

The Gradient Descent Algorithm 3.1 may be seen as an instance of the General Descent Method Algorithm 3.2 by setting the search direction to the negative gradient of the cost function $J(\boldsymbol{\theta})$,

$$\Delta\boldsymbol{\theta} := -\nabla J(\boldsymbol{\theta}),$$

and specifying that the line search method is an **exact line search** in the sense that the choice of the hyperparameter α is selected along the ray

$$\{\boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta} : \alpha > 0\}$$

to minimize the cost function J.

Most descent methods do not use an exact line search because of the computational cost of computing an exact minimization problem *for each iteration* of the algorithm until the stopping condition is met. Exact line search is an efficient choice only if the cost of minimizing J with $\boldsymbol{\theta}$ is comparably lower than the cost of computing the search direction alone. However, in general this is not the case: typically finding the direction is easy and solving exact minimization problems is considerably harder. This fact suggests that an *inexact* line search method ought to be used instead. **Backtracking lines search** is a common, approximate line search method.

Backtracking Line Search

Algorithm 3.3.

```
1: Take as input a descent direction $\Delta\boldsymbol{\theta}$
2: Choose parameters $a \in (0, 0.5)$ and $b \in (0, 1)$.
3: $\alpha = 1$
4: while $J(\boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}) > J(\boldsymbol{\theta}) + \alpha a\nabla J(\boldsymbol{\theta})^T\Delta\boldsymbol{\theta}$ do
5:     $\alpha := b\alpha$
6: end while
```

The Backtracking Line Search Algorithm 3.3 starts by taking a 1 unit set, $\alpha = 1$, then reduces the step size by the factor a strictly between 0 and 1. Intuitively, backtracking line search takes a gigantic first step in the direction of the (negative) gradient, then progressively smaller steps with each iteration. The idea is that, while the first few steps are likely to be considerably more inaccurate than exact line search, the iterative process of the algorithm allows for quick corrections. Coupled with a decaying step size, backtracking line search can often quickly move to the general region where a local minimia exists and then converge much as exact line search would do.

3.3 Statistical Gradient Descent*

Gradient descent may be slow to compute, particularly for large data sets. Why? Because for each step of the gradient, we have to look at all m training examples. For very large data sets, m where m is extremely large, this can be impractical.

One way around this problem is to **estimate the gradient** by taking random samples from your training data. Gradient descent is an iterative procedure, so intuitively we have many opportunities to recover from minor errors from estimates of the gradient. This trade-off between a quick but approximate estimate of the gradient, with many opportunities to make corrections, versus a slow but precise gradient value is a type of trade-off that comes up often in machine learning, where often a quick but noisy procedure outperforms a slow but more precise procedure. In fact, sometimes 'slow and precise' procedures are computationally intractable leaving no choice but a 'quick and noisy' approach.

Instead of computing the gradient exactly by looking at the **full batch** of

$$\nabla \hat{L}_m(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}}(x^{(i)}), y^{(i)})$$

for a training set $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots, (x^{(m)}, y^{(m)})\}$, take a subset $\mathcal{S} \subseteq \mathcal{D}$ of size $M < m$, which we may write using the index convention m_1 to m_M, or $\mathcal{S} = \{(x^{(m_1)}, y^{(m_1)}), \dots, (x^{(m_M)}, y^{(m_M)})\}$. Then compute the gradient of this **minibatch** of M training examples randomly sampled from m by

$$\nabla \hat{L}_M(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}}(x^{(m_i)}), y^{(m_i)})$$

This minibatch gradient is an **unbiased estimator** for the full batch gradient, that is

$$\mathbb{E}\left[\nabla \hat{L}_M(\boldsymbol{\theta})\right] = \nabla \hat{L}_m(\boldsymbol{\theta}) \tag{3.4}$$

Note that the bigger the minibatch, that is, the greater the size M of the sample $\mathcal{S}$ from the training set $\mathcal{D}$, the better the estimate. For this to work it is necessary that the estimator be unbiased, thus it is crucial that the minibatches are **random samples** drawn from the training data.

In general, there is a trade-off in minibatch size where a larger M yields a better estimate of the gradient but is slower to compute, and a smaller M yields a worse estimate but can be very fast. How small can M be? We can get good results taking a single data point, $M = 1$! This is called **stochastic gradient descent** (SGD).

The interesting point to highlight is that a single data point estimate yields a terrible estimate of the gradient, as bad an estimate as random sampling could give you. But, the estimate is unbiased (because it is random), so accumulating several of them "averages out" the errors in the estimates.

Minibatch gradient descent uses random subsets of size M, where M is typically between 1 and a few hundred, where $M = 32$ is a good default value.[2] With $M \geq 10$, you get speedup (per data point).

What about getting stuck in local minima? This may help you get to a good local minima (for non-convex functions) by introducing random noise to help you get out of a path leading to a sub-optimal local minimum.[3]

[2]See Yoshua Bengio's "Practical recommendations for gradient-based training of deep architectures"

[3]See (Bottou 2012). See also (Boyd and Vandenberghe 2004, Chapter 9)

Chapter 4

Classification

A **classification problem** is a type of supervised learning problem where input features $\boldsymbol{x}$ are used to predict a discrete category label y. For example, you might look at housing data to predict whether a house will sell ($y = 1$) or not ($y = 0$) within 30 days of going on the market. This is an instance of a **binary classification** problem, where the response variable y typically takes values from the set $\{0, 1\}$. Rather than merely predict a classification label, we might instead wish to model **how likely** it is that a response is positive, $y = 1$, given features $\boldsymbol{x}$. Put differently, classification algorithms invariably seek to model the probability of a class label as some function of features $\boldsymbol{x}$.

4.1 Logistic Regression

A **classification problem** is a type of supervised learning problem where input features $\boldsymbol{x}$ are used to predict a discrete category label y. For example, you might look at housing data to predict whether a house will sell ($y = 1$) or not ($y = 0$) within 30 days of going on the market. This is an instance of a **binary classification** problem, where the target variable y typically takes values from the set $\{0, 1\}$.

Rather than merely predict a classification label, we instead wish to model **how likely** it is that a response is positive, $y = 1$, given features $\boldsymbol{x}$. Put differently, classification algorithms invariably seek to model the probability of a class label as some function of features $\boldsymbol{x}$:

$$\begin{aligned} p(y = 1|x; \boldsymbol{\theta}) &= h(x; \boldsymbol{\theta}) \\ p(y = 0|x; \boldsymbol{\theta}) &= 1 - h(x; \boldsymbol{\theta}) \end{aligned}$$

which can be rewritten as

$$p(y^{(i)}|x^{(i)}; \boldsymbol{\theta}) = \left(h(x^{(i)}; \boldsymbol{\theta})\right)^{y^{(i)}} \left(1 - h(x^{(i)}; \boldsymbol{\theta})\right)^{1-y^{(i)}} \tag{4.1}$$

where for each i, either $y^{(i)} = 0$ or $y^{(i)} = 1$.

A very popular classification model is **logistic regression** which, like linear regression, is linear in the parameters $\boldsymbol{\theta}$. Indeed, as we will see, logistic regression may be viewed as an extension of linear regression: this is the reason the name ‘regression’ appears, even though logistic regression is solely a classification model.

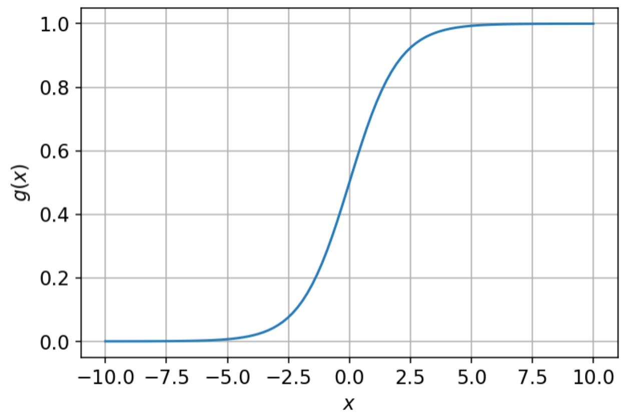


Figure 4.1: Sigmoid function

Specifically, we can think of logistic regression as making two changes to linear regression: (i) replace the Gaussian distribution for y, Equation ??, with a **Bernoulli** distribution, for a binary response variable $y \in \{0, 1\}$; and (ii) restrict the range of values of $h_{\boldsymbol{\theta}}$ to the unit interval, [0,1], by passing the computed value of the linear combination parameters and features, $\boldsymbol{\theta}^T \boldsymbol{x}$, to a function $g(\cdot)$:

$$y^{(i)}|x^{(i)} = x^{(i)} \sim \text{Bernoulli}(g(\boldsymbol{x})) \tag{4.2}$$

$$P(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \text{Bernoulli}(y \mid g(\boldsymbol{x}))$$

where $g(\boldsymbol{x}) = \mathbb{E}[y \mid \boldsymbol{x}] = P(y = 1 \mid \boldsymbol{x})$ is the conditional expectation of y being in the positive class, $y = 1$, given the observed features $\boldsymbol{x}$ and parameter vector $\boldsymbol{\theta}$, and $0 \leq g(\boldsymbol{x}) \leq 1$ is the **sigmoid function**:

$$g(\boldsymbol{x}) = \text{sigm}(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \tag{4.3}$$

The sigmoid function is a special case of the **logistic** or **logit** function, hence the name ‘logistic regression’. For a k-dimensional covariate, X, the conditional probability $p(y^{(i)} = 1|X = x) = q_i$, the **logit function** is

$$\text{logit}(\boldsymbol{q}_i) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \tag{4.4}$$

The sigmoid $\text{sigm}(\infty) = 1, \text{sigm}(-\infty) = 0$, and $\text{sigm}(0) = 0.5$.

Logistic regression is a popular model for classification for several reasons. First, logistic regression models are relatively easy and fast to implement. Specifically, there are implementations of logistic regression which are linear in the number of non-zeros in the data set. In contrast, more complicated classifiers like support vector machines, neural nets, and decision trees are often much slower to train.

Second, the models are easy to interpret in terms of **odds ratios**. The odds ratio, $q/1-q$, by convention, compares the probability of success (q) to the probability of failure ($1 - q$) in a binomial experiment. If the probability of winning a prize is 0.75, then we say the odds ratio is $^{.75}/_{.25} = 3$ and therefore that the odds of winning are 3 to 1. The corresponding probability of success in a logistical model is computed by

$$\log p = \frac{e^{\theta_0+\theta_1 \boldsymbol{x}}}{1+e^{\theta_0+\theta_1 \boldsymbol{x}}} = \frac{\boldsymbol{\theta}^T\boldsymbol{x}}{1+\boldsymbol{\theta}^T\boldsymbol{x}}$$

so the logistic odds of success are

$$\frac{q}{1-q} = e^{\theta_0+\theta_1 \boldsymbol{x}} = e^{\boldsymbol{\theta}^T\boldsymbol{x}}$$

Equation 4.5 then defines the **log odds** ratio by taking the log of each side:

$$\log \frac{p(y=1|\boldsymbol{x};\boldsymbol{\theta})}{p(y=0|\boldsymbol{x};\boldsymbol{\theta})} = \log \frac{q}{1-q} = \boldsymbol{\theta}^T\boldsymbol{x} \tag{4.5}$$

which is a linear function of $\boldsymbol{\theta}$. Yet the interpretation is the same: the higher the value of Equation 4.5 corresponds to a higher probability of success, i.e., a higher probability that $y = 1$. Because the conditional probabilities depend on the linear combination features *and* parameters, you often see the conditional probabilities appearing in 4.5 include the parameter vector $\boldsymbol{\theta}$ as well as the feature vector $\boldsymbol{x}$.

A third reason for the popularity of logistic regression models is that they are easy to extend to multi-class classification, and easy to extend to handle non-linear decision boundaries. We will return to these two points below.

4.2 Cross Entropy and Negative Log Likelihood

Although there are other smoothing functions, the sigmoid function is used in part because its first derivative, $g'(z)$, has a useful form:

$$\begin{aligned} g'(z) &= \frac{d}{dx}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}\cdot e^{-z} \\ &= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\ &= g(z)\,(1-g(z)) \end{aligned} \tag{4.6}$$

where in our application of the sigmoid function to logistic regression $z = \boldsymbol{\theta}^T\boldsymbol{x}$.

To fit $\boldsymbol{\theta}$ to a logistic regression model, we shall adapt the (negative) log likelihood strategy argument described in Section **??**. The reasoning is the same as before. In Section **??** we saw how least squares regression is derived as the *maximum* likelihood estimator under a specific set of assumptions. In this section we specify the probabilistic assumption necessary to estimate the parameters of a logistic regression as those which maximize (minimize) log (negative log) likelihood.

First, following convention, we assume that the probability estimate of the model is for the positive class $y = 1$. In other words, the hypothesis for logistic regression is

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := P(y = 1 \mid \boldsymbol{x}, \boldsymbol{\theta})$$

and $P(y = 0 \mid \boldsymbol{x}, \boldsymbol{\theta})$ is simply $1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})$. Since the binary response variable y in logistic regression is modeled by a Bernoulli distribution, we may represent the hypothesis for positive and negative classes by

$$P(y \mid \boldsymbol{x}, \boldsymbol{\theta}) := h_{\boldsymbol{\theta}}(\boldsymbol{x})^y (1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^{1-y}, \quad \text{where } y \in \{0, 1\} \tag{4.7}$$

If the m training examples in our training data were generated independently, then the likelihood of the parameters factorizes:

$$\begin{aligned} L(\boldsymbol{\theta}) &= P(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{m} P\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \boldsymbol{\theta}\right) \\ &= \prod_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)^{y^{(i)}} \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)^{1-y} \end{aligned}$$

where $\boldsymbol{X}$ is an $m \times n$ matrix for n-features. Then, the **log likelihood** is given by

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{m} y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))$$

Finally, just as we changed signs to minimize log-likelihood to compute least mean squared error in Equation **??** to fit linear regression, there is a corresponding negative log-likelihood for logistic regression which is called the **cross-entropy** error function:

$$NLL(\boldsymbol{\theta}) := -\sum_{i=1}^{m} y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \tag{4.8}$$

Turn now to the minimization of the function NNL and, to cut down on indices, let's consider a single training example (x, y). The partial derivative for θ_j of $\boldsymbol{\theta}$ is derived by

$$\begin{aligned} \frac{\partial}{\partial \theta_i} NLL(\boldsymbol{\theta}) &= \left(y \frac{1}{g(\boldsymbol{\theta}^T \boldsymbol{x})} - (1 - y) \frac{1}{1 - g(\boldsymbol{\theta}^T \boldsymbol{x})}\right) \frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^T \boldsymbol{x}) \\ &= \left(y \frac{1}{g(\boldsymbol{\theta}^T \boldsymbol{x})} - (1 - y) \frac{1}{1 - g(\boldsymbol{\theta}^T \boldsymbol{x})}\right) g(\boldsymbol{\theta}^T \boldsymbol{x})(1 - g(\boldsymbol{\theta}^T \boldsymbol{x})) \frac{\partial}{\partial \theta_j} \boldsymbol{\theta}^T \boldsymbol{x} \\ &= \left(y \left(g(\boldsymbol{\theta}^T \boldsymbol{x})\right) - (1 - y) \left(1 - g(\boldsymbol{\theta})^T \boldsymbol{x}\right)\right) \quad \text{(by Equation 4.6)} \\ &= (y - h_{\boldsymbol{\theta}}(\boldsymbol{x})) x_j \end{aligned}$$

This gives us the gradient descent rule,

$$\theta_i := \theta_i - \alpha\left(h(x^{(k)}) - y^{(k)}\right) x_i^{(k)}$$

which we may generalize for all m training examples:

$$\theta_i := \theta_i - \alpha\frac{1}{m}\sum_{j=1}^{m}(h_\theta(x^{(j)}) - y^{(j)})x^{(j)} \tag{4.9}$$

Compare Equation 4.9 to Equation 3.3. Although the have the same form, the update rules are not identical. Note that $h_{\boldsymbol{\theta}}(\cdot)$ refers to two different hypotheses. Cross-entropy,

$$\begin{aligned}\log L(\boldsymbol{\theta}) &= \sum_{i=1}^{m} y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))\\ &= \sum_{i=1}^{m} y^{(i)} \log \operatorname{sigm}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \operatorname{sigm}(\boldsymbol{x}^{(i)}))\end{aligned}$$

is not equivalent to the residual sum of squared errors. In fact, unlike linear regression, we cannot write down a closed-form solution to Equation 4.8.

4.3 Generalized Linear Models

In Chapters 2 and 3 we looked at linear regression, with the distribution $y \mid \boldsymbol{\theta} \sim \mathcal{N}(\mu, \sigma^2)$, and in this chapter we looked at logistic regression with the distribution $y \mid x, \boldsymbol{\theta} \sim \text{Bernoulli}(p)$ for appropriate choice of μ and p as a function of features $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$. We also saw that they shared the form of a common update rule. This was not a coincidence.

Both linear regression and logistic regression belong to a broader family of models called **Generalized Linear Models** (**GLM**). By looking at GLMs, we can show how other models in this family can be derived and applied to classification and regression problems.[1]

The Exponential Family

A first step to defining GLMs is to define a broader family of distributions, called the **exponential family** of distributions, to which the Gaussian and Bernoulli belong.

[1]This section is adapted from Andrew Ng's Stanford CS229 lecture notes.

Exponential Family

Definition 4.1. A probability distribution $P(y;\eta)$ belongs to the exponential family if

$$P(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

where:

- η is the **natural** or **canonical parameter** of the distribution;
- $T(y)$ is the **sufficient statistic** for y;
- $a(\eta)$ is the **log partition function**;

and $e^{-a(\eta)}$ acts as a normalizing constant to ensure the distribution $P(y,\eta)$ sums or integrates over y to 1.

Choosing T, a and b defines a family of distributions parameterized by η. Varying η while keeping T, a and b fixed yields different distributions within a specific family. For the distributions we will consider, the sufficient statistic is simply $T(y) = y$.

To show that the Bernoulli distribution is a member the exponential family, observe that the Bernoulli distribution with mean p, written $\mathrm{Bernoulli}(p)$, specifies a probability distribution over $y \in \{0,1\}$ such that $P(y=1;p) = p$ and $P(y=0;p) = 1-p$. Changing values of p yields different Bernoulli distributions with different means—that is, changing p yields different Bernoulli distributions with different probabilities of success, $y = 1$.

The Bernoulli distribution is written as:

$$\begin{aligned} P(y;p) &= p^y(1-p)^{1-y} \\ &= \exp(y\log p + (1-y)\log(1-p) \\ &= \exp\left(\left(\log\left[\frac{p}{1-p}\right]\right)y + \log(1-p)\right) \end{aligned}$$

So, for logistic regression, the natural parameter is $\eta = \log\left({}^{p}/_{1-p}\right)$. Observe that if we invert this definition for η and instead solve for p in terms of η, then we derive the sigmoid function: $p = {}^{1}/_{1+e^{-\eta}}$ To complete the specification of the Bernoulli distribution as an exponential family distribution, we have:

$$\begin{aligned} T(y) &= y \\ a(\eta) &= -\log(1-\phi) = \log(1+e^{\eta}) \\ b(y) &= 1 \end{aligned}$$

Turning to the Gaussian distribution, recall that σ^2 did not have an effect on the choice of $\boldsymbol{\theta}$ with respect to $h_{\boldsymbol{\theta}}(x)$. To simplify the derivation, let $\sigma^2 = 1$. Then,

$$P(y;p) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(y-\mu)^2\right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \cdot \exp\left(\mu y - \frac{1}{2}\mu^2\right)$$

For the Gaussian distribution, the natural parameter is $\eta = \mu$. To complete the specification of the Gaussian distribution as an exponential family distribution we have:

$$\begin{aligned} T(y) &= y \\ a(\eta) &= \frac{\eta^2}{2} = \frac{\mu^2}{2} \\ b(y) &= \left(\frac{1}{\sqrt{2\pi}}\right) \exp\left(\frac{-y^2}{2}\right) \end{aligned}$$

In addition to the Bernoulli and Gaussian distributions, the multinomial, Poisson, gamma, exponential, beta, and Dirichlet distributions are also members of the exponential family—and many more. Apart from constructing generalized linear models, which we discuss next, the exponential family is used in other important methods to **integrate heterogenous data** into a single statistical model.

It is increasingly common to collect heterogenous data about a given set of subjects, often from multiple *types* of measurements. In addition to continuous data and binary-valued data, which we have seen in our presentation of linear regression and logistic regression, respectively, one might have count data, text data, and geo-location data. Since there are exponential family members that are appropriate for handling most of the different data types one finds in heterogenous dataset, one approach to constructing probabilistic models to *jointly* model multiple data-types is to use the exponential family to construct closed-form, joint multivariate graphical models.

Although data integration is beyond the scope of this chapter, we mention it as an additional reason to familiarize yourself with the exponential family of distributions.

How to Derive a GLM

Suppose you are faced with either a regression or classification problem involving the prediction of a random variable y as a function of a (vector of) random variables $\boldsymbol{x}$. To derive a GLM for such a problem, three assumptions are made:

1. **Exponential Family.** Given $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$, the distribution of y is assumed to follow some exponential family distribution with natural parameter η, written $y \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \text{ExponentialFamily}(\eta)$.
2. **Expected Value of $\boldsymbol{T(y)}$.** Given features $\boldsymbol{x}$, the goal is predict the expected value of the sufficient statistic $T(y)$. When $T(y) = y$, this says that the prediction $h(\boldsymbol{x})$ ought to satisfy $\mathbb{E}[y \mid \boldsymbol{x}]$.
3. **Linearity.** The natural parameter η and the input features $\boldsymbol{x}$ are linearly related: $\eta = \boldsymbol{\theta}^T \boldsymbol{x}$

Suppose the target variable y is continuous and the conditional distribution of y given features $\boldsymbol{x}$ is Gaussian, $\mathcal{N}(\mu, \sigma^2)$. The Gaussian distribution is a member of the Exponential Family when η is the set to the mean, μ. (Recall that we set $\sigma^2 = 1$ because variance does not play a role in fitting $\boldsymbol{\theta}$ in minimizing squared error loss.) Given the GLM parameterization for a Gaussian distribution from above, we may derive as a

$$\begin{aligned}
h_{\boldsymbol{\theta}} &= \mathbb{E}\left[y \mid \boldsymbol{x}, \boldsymbol{\theta}\right] && \text{Assumption 2} \\
&= \mu && \text{The expected value of } y|\boldsymbol{x}, \boldsymbol{\theta} \sim \mathcal{N}(\mu, \sigma^2) \text{ is } \mu \\
&= \eta && \text{Assumption 1 and parameterization of the} \\
& && \text{Gaussian distribution by } \mathrm{ExponentialFamily}(\eta) \\
&= \boldsymbol{\theta}^T \boldsymbol{x} && \text{By Assumption 3}
\end{aligned}$$

Logistic Regression as a GLM

Suppose the target variable y is binary-valued, $y \in \{0, 1\}$, and the conditional distribution of y given $\boldsymbol{x}$ is a Bernoulli distribution, $y \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \mathrm{Bernoulli}(p)$. The GLM parameterization of the Bernoulli distribution, recall, is $\mathrm{ExponentialFamily}(\eta)$ when $p = {}^{1}\!/_{1+e^{-\eta}}$. Therefore,

$$\begin{aligned}
h_{\boldsymbol{\theta}} &= \mathbb{E}\left[y \mid \boldsymbol{x}, \boldsymbol{\theta}\right] && \text{Assumption 2} \\
&= p && \text{The expected value of } y = 1|\boldsymbol{x}, \boldsymbol{\theta} \sim \mathrm{Bernoulli}(p) \text{ is } p \\
&= \tfrac{1}{1+e^{-\eta}} && \text{Assumption 1 and parameterization of the} \\
& && \text{Bernoulli distribution by } \mathrm{ExponentialFamily}(\eta) \\
&= \tfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} && \text{By Assumption 3}
\end{aligned}$$

Now we have an answer to the question of why, for logistic regression, does the hypothesis $h_{\boldsymbol{\theta}}(x)$ have the form $\frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$: if we assume that y conditional on x is a Bernoulli distribution, this form follows from the definition of GLMs and the properties of exponential family distributions.

One last thing. The sigmoid function $g(\eta)$ of the natural parameter η gives the distribution mean (i.e., the probability of success). In general, we call the function $g(\cdot)$ giving the distribution mean as a function of η the **canonical response function**, and the inverse g^{-1} is called the **canonical link function**. The canonical response function for logistic regression is the sigmoid logistic function. The canonical response function for the Gaussian family is simply the identity function.

4.4 Generative vs Discriminative Models

If so, we may wish to construct a probabilistic classifier, and there are two general routes one can take.

The first approach to creating a probabilistic classifier, called a **generative classifier**, creates a joint probability model $P(y, \boldsymbol{x}) = P(y)P(\boldsymbol{x}|y)$, where $P(y)$ by convention is understood as the probability that $y = 1$. With this joint probability model, and observed features $\boldsymbol{x}$, the generative classifier derives the posterior probability $P(y|\boldsymbol{x})$. A second approach, called a **discriminative classifier**, is to directly fit a model of the form $P(y|x)$ to map the input feature vector $\boldsymbol{x}$ to the output response variable y.

The principal difference between these two models is how they are trained. Each typically seeks to maximize log likelihood—a point we will return to shortly. A key difference, however, is that generative classifiers are usually fit by maximizing log likelihood of the **joint** distribution $P(y, \boldsymbol{x})$, whereas discriminative classifiers are fit by maximizing log likelihood of the **conditional** distribution that follow this second approach $P(y|\boldsymbol{x})$. There are consequences which follow from each, leading to trade-offs. For instance, A generative model tends to make strong independence assumptions that allow one to "factorize" the joint distribution into separable, probabilistically independent components. This technique underpins **Bayesian networks** (Spirtes, Glymour, and Scheines 2000; Haenni, Romeijn, Wheeler, and Williamson 2011), for instance. A popular example of a generative classifier is **naive Bayes**.

MODEL	CLASSIF/REGR/BOTH	GEN/DISCR	PARAM/NON-PARAM
Discriminant Analysis	Classif	Gen	Param
Naive Bayes Classifier	Classif	Gen	Param
Linear Regression	Regr	Discr	Param
Logistic Regression	Classif	Discr	Param
Neural Networks	Both	Discr	Param
K-nearest neighbor classifier	Classif	Gen	Non
CART	Both	Discr	Non
Boosted Model	Both	Discr	Non
Support Vector Machine	Both	Discr	Non
Gaussian processes (GP)	Both	Discr	Non

Table 4.1: Some Common Classification and Regression Models

It is helpful to know whether a model is generative or discriminative, just as it is helpful to know whether a model is **parametric** or **parametric**, and whether the model can be used for classification problems, regression problems, or both. For instance, using these properties, a number of classification and regression models can be conveniently categorized in Table 4.1, adapted from (Murphy 2012, p. 273).

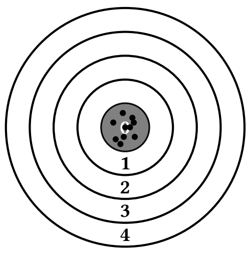
Chapter 5

Learning Theory

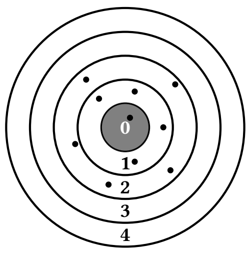
When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind. –William Thomson, Lord Kelvin, 1889

Men and instruments will blunder
Calculating things of wonder. –A. M. Sullivan, 1970

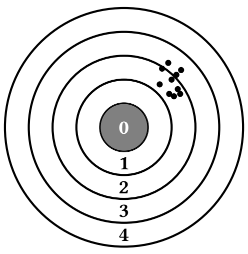
Suppose your task is to select a pea-shooter from four on offer, each distinguished by the central tendency (bias) and dispersion (variance) of peas shot at a target.



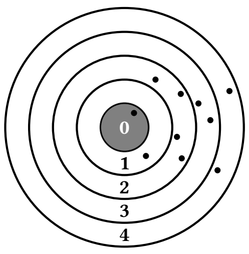
low bias & low variance



low bias & high variance



high bias & low variance



high bias & high variance

If the goal is to minimize your score, the first pea-shooter is your best choice since it has both a low bias and low variance. Absent this option, and given the choice between a low bias and high variance pea-shooter versus a high bias and low variance pea shooter, the second shooter's score (14) is lower than the third's (28), so the second pea-shooter is the next best choice.

low bias and high variance

, and the **bias-variance trade-off** refers to a particular decomposition of error that sometimes allows for the reduction in overall error by increasing bias to afford a comparatively larger reduction in variance, or vice versa.

Ideally, you would prefer a procedure for delivering your "shots" that had both a low bias and low variance. Absent that, and given the choice between a low bias and high variance procedure versus a high bias and low variance procedure, you would presumably prefer the latter procedure if it returned a lower overall score than the former, which is true of the corresponding figures above. Although a a learning algorithm ideally will have low bias and low variance, in practice it is common that the reduction in one type of error yields some increase in the other.

5.1 What is a Learning Problem?

Recall that in Section 1.1 we presented the notion of a learning algorithm in terms of a task to be performed, the experience the algorithm is designed to have of the data, and a numeric performance measure. This definition has its roots in statistical learning theory. In this chapter we return to the question of what a learning problem is, focusing on supervised learning. Statistical learning theory conceives of a supervised learning problem in terms of the *input* to, and *output* from, a learner, and a *performance measure.*

Input. The experience that a (supervised) machine learning algorithm has of data—or more generally, the experience the *learner* has to learn from—may be understood in terms of a domain of objects, a set of labels the learner wishes to assign to those objects, and a sample of labeled observations of that domain:

1. a **domain** of objects $\mathcal{X}$ the learner wishes to label, where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$.
2. a **label set** $\mathcal{Y}$ of labels:
   - *Regression:* $\mathcal{Y} = \mathbb{R}$
   - *Classification:* $\mathcal{Y} = \{0, 1\}$
3. a **training set** $\mathcal{D} = [(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)})]$ is a finite list of *labeled observations* of the domain, that is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$.

Output. The learner is given the task of producing a **prediction rule** or **hypothesis** h that maps objects in the domain to labels, that is $h : \mathcal{X} \mapsto \mathcal{Y}$. Notice that prediction rules apply to the domain of objects, not to the training set the learner is exposed to. A paradigmatic use of a prediction rule is to predict the label of a new (i.e., unseen) object.

Performance Measure. A **performance measure** or **loss function** ℓ measures the error of the prediction made by the rule h, given $\boldsymbol{x}$, by comparing $h(\boldsymbol{x})$ to applied to the observation y, written $\ell\,(\boldsymbol{x}, y, h(\boldsymbol{x}))$ where $\ell : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \mapsto [0, \infty)$. It is common to use the shorthand $\ell\,(y, h(\boldsymbol{x}))$ or $\ell\,(h(\boldsymbol{x}), y)$, since $\boldsymbol{x}$ is implicit.

Viewing supervised learning problems in learning theoretic terms will help to abstract away from the material we have covered so far to accommodate the learning algorithms that will come in future chapters.

5.2 Loss Functions

At the heart of most machine learning algorithms is a method, or methods, for solving numerical optimization problems. An **optimization problem** involves maximizing or minimizing a function, $f(\boldsymbol{\theta})$, called the **objective function**, by changing the values of $\boldsymbol{\theta}$. When the task is to minimize the objective function on the range $[0, \infty)$, the goal is to make f as close to 0 as possible. We refer to this specific type of objective function as a **cost function** and use the notation $J(\boldsymbol{\theta})$, instead of a generic f, to represent cost functions.

Every cost function is constructed in terms of a **loss function**, which measures the compatibility of $h(\boldsymbol{x})$ with the true state y. Recall that for univariate linear regression, defined by Equation **??**, the vector $\boldsymbol{\theta} = [\theta_1, \theta_2]$ comprises the two **regression coefficients**, which are the adjustable parameters of the model. The cost function $J(\boldsymbol{\theta})$, as defined by Equation 2.23, is then used for evaluating the fit of the regression model $h(\boldsymbol{x})$ to the set of m training examples.

We discussed two methods for finding values for the vector $\boldsymbol{\theta}$ that minimize the ordinary least squares equation, defined by Equation 2.3: the closed form solution provided by the normal equation, Equation **??**, and an approximate solution that uses the general optimization algorithm Gradient Descent, Algorithm 3.1.

The loss function underpinning OLS regression is the **squared-loss** function, which is also called **ℓ_2-loss**. We introduced squared loss in section 2.5, but we rewrite it here in sightly different notation:[1]

$$\ell_2\left(y, h(\boldsymbol{x})\right) = \left(h(\boldsymbol{x}) - y\right)^2 \tag{2.21}$$

But ℓ_2-loss is not the only way to measure the difference between a prediction and the true label value. For instance, we might instead measure the **linear loss**:

$$\ell_1\left(y, h(\boldsymbol{x})\right) = |h(\boldsymbol{x}) - y| \tag{5.1}$$

A well-known issue with ℓ_2-loss is that it is sensitive to outliers. Figure 5.1 illustrates the comparative influence of squared loss and linear loss by fitting a single data set by ordinary least squares (OLS) and by **least absolute difference** (LAD), respectively.

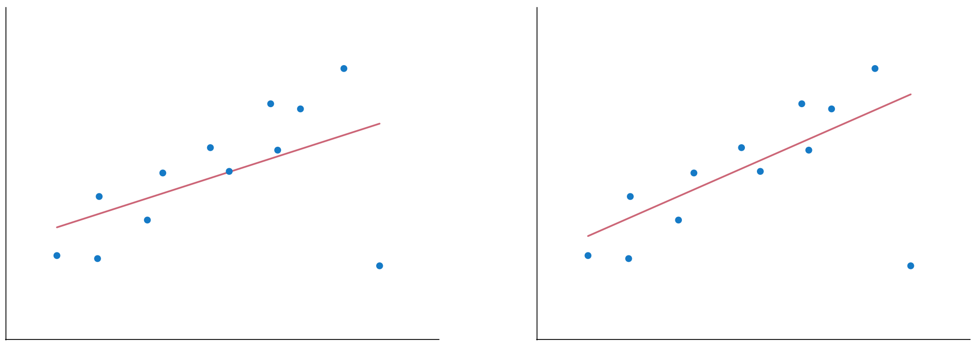


Figure 5.1: Left: OLS Regression fit with ℓ_2-loss; Right: LAD Regression fit with ℓ_1-loss.

These two loss functions are connected to a surprisingly wide range of important theoretical concepts. For instance, although we have focused on choosing parameters $\boldsymbol{\theta}$ to minimize $J(\boldsymbol{\theta})$, but we can reframe this task more generally as a statistical **decision problem**—that is, as a game you play against nature (Berger 1985). In this game nature selects the target label y, which is unknown to you, and nature also produces an observation, $\boldsymbol{x}$, that is shown to you. You then have to choose an "act", h, that is

[1]Specifically, we replace the scalar x with a feature vector $\boldsymbol{x}$ and omit the index i, writing $\boldsymbol{x}$ and y instead of $\boldsymbol{x}^{(i)}$ and $y^{(i)}$.

an optimal response (prediction) to each possible observation, where a hypothesis h is optimal just in case it minimizes the *expected loss*:

$$\arg\min_h \mathbb{E}\left[\ell(y,h)\right] \tag{5.2}$$

Students of economics will recognize Equation 5.2 as a variant of the **expected utility hypothesis** by substituting the loss incurred by h when the state is y for the negative loss, which is called the **utility function**, u:

$$-\ell(y,h) = u(y,h)$$

and switching the optimization task in Equation 5.2 from minimizing ℓ to maximizing u. Then it can be shown analytically that the optimal strategy in this game for estimating the unknown y with respect to ℓ_2-loss is to choose h to align with the mean of the observations. Similarly, it can be shown analytically that the optimal estimate of unknown y with respect to ℓ_1-loss is to align h with the median of the observations.

Whereas Figure 5.1, left, depicts the disproportionate influence that a single outlying data point has on OLS regression with ℓ_2-loss, LAD regression with ℓ_1-loss is subject to a different form of instability arising from *small* variations in data—such as those that can arise from small rounding errors. In some cases, a small movement of data has a very large effect on the fit of an LAD regression model (Ellis 1998). In short, whereas OLS regression is sensitive to outliers but robust to small movements of data, LAD regression is robust to outliers but *can* be unstable to small movements of data.

Another difference between these two loss functions is that the gradient of ℓ_1-loss, unlike the gradient of ℓ_2-loss, does not exist at 0. Moreover, since the gradient of ℓ_1-loss is constant, Gradient Descent's fixed step-size renders the algorithm impractical to implement for ℓ_1-loss without significant modifications.

There are alternatives to ℓ_1-loss and ℓ_2-loss which attempt to combine several of the nice mathematical properties of ℓ_2-loss with robustness properties similar to those enjoyed by ℓ_1-loss. One approach, Huber's **M-loss** (Huber 1964), specifies a family of intermediaries between ℓ_1-loss and ℓ_2-loss, one of which is

$$\ell\left(y, h(\boldsymbol{x})\right) = \begin{cases} \frac{1}{2}\left(y - h(\boldsymbol{x})\right)^2 & \text{if } |y - h(\boldsymbol{x}| < 1 \\ |y - h(\boldsymbol{x})| - \frac{1}{2} & \text{otherwise} \end{cases} \tag{5.3}$$

The full theory of M-estimators and their associated loss functions will not concern us here. Hence, we shall refer to Equation 5.3 simply as **Huber's robust loss**.

Figure 5.2 plots ℓ_1-loss, ℓ_2-loss and Huber's robust loss for residual errors ranging from -3 to 3.

5.3 Risk

Although optimization methods are integral to machine learning, their use for training machine learning models is different from their use in solving straightforward optimization problems. The main difference is that we use the methods of numerical

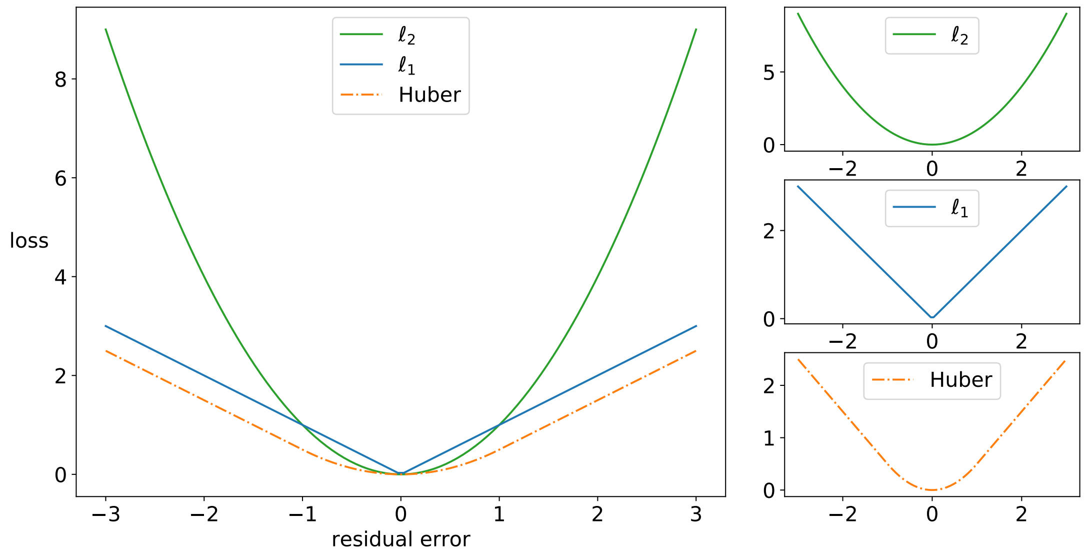


Figure 5.2: ℓ_1-loss, ℓ_2-loss, and Huber's robust loss.

optimization on training data, but our aim is to make accurate predictions on observations we have not trained on. As will become clear, minimizing an objective function defined with respect to your training data does not entail that the same objective function is minimized with respect to test data consisting of new observations. Fitting and predicting are two very different things.

So far we have written cost functions with respect to the training data

$$\mathcal{D} = [(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)})],$$

which we may characterize by the **empirical probability distribution** $\hat{p}_{\mathcal{D}}$ such that, trivially, $(\boldsymbol{x}, y) \sim \hat{p}_{\mathcal{D}}$.

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\hat{p}}[\ell(y, h_{\boldsymbol{\theta}}(\boldsymbol{x}))] \tag{5.4}$$

5.4 The Bias-Variance Decomposition of Mean Squared Error

Predicting the exact volume of gelato to be consumed in Rome next summer is more difficult than predicting that more gelato will be consumed next summer than next winter. For although it is a foregone conclusion that higher temperatures beget higher demand for gelato, the precise relationship between daily temperatures in Rome and *consumo di gelato* is far from certain. Modeling quantitative, predictive relationships between random variables, such as the relationship between the temperature in Rome, X, and volume of Roman gelato consumption, Y, is the subject of *regression analysis.*

Suppose we predict that the value of Y is h. How should we evaluate whether this prediction is any good? Intuitively, the best we can do is to pick an h that is as close to Y as we can make it, one that would minimize the difference $Y - h$. If we are indifferent to the direction of our errors, viewing positive errors of a particular magnitude to be no worse than negative errors of the same magnitude, and vice versa, then a common practice is to measure the performance of h by its squared difference from Y, $(Y - h)^2$.

(We are not always indifferent; consider the plight of William Tell aiming at that apple.) Finally, since the values of Y vary, we might be interested in the average value of $(Y-h)^2$ by computing its expectation, $\mathbb{E}\left[(Y-h)^2\right]$. This quantity is the *mean squared error* of h,

$$\mathrm{MSE}(h) := \mathbb{E}\left[(Y-h)^2\right].$$

Now imagine our prediction of Y is based on some data $\mathcal{D}$ about the relationship between X and Y, such as last year's daily temperatures and daily total sales of gelato in Rome. The role that this particular dataset $\mathcal{D}$ plays as opposed to some other possible data set is a detail that will figure later. For now, view our prediction of Y as some function of X, written $h(X)$. Here again we wish to pick an $h(\cdot)$ to minimize $\mathbb{E}\left[(Y-h(X))^2\right]$, but how close $h(\cdot)$ is to Y will depend on the possible values of X, which we can represent by the conditional expectation

$$\mathbb{E}\left[(Y-h(X))^2\right] := \mathbb{E}\left[\mathbb{E}\left[Y-h(X) \mid X\right]\right].$$

How then should we evaluate this conditional prediction? The same as before, only now accounting for X. For each possible value x of X, the best prediction of Y is the conditional mean, $\mathbb{E}\left[Y \mid X=x\right]$. The *regression function* of Y on X, $r(x)$, gives the optimal value of Y for each value $x \in X$:

$$r(x) := \mathbb{E}\left[Y \mid X=x\right].$$

Although the regression function represents the true population value of Y given X, this function is usually unknown, typically complicated, therefore often approximated by a simplified model or learning algorithm, $h(\cdot)$.

We might restrict candidates for $h(X)$ to linear (or affine) functions of X, for instance. Yet making predictions about the value of Y with a simplified linear model, or some other simplified model, can introduce a systematic prediction error called *bias*. Bias results from a difference between the central tendency of data generated by the true model, $r(X)$ (for all $x \in X$), and the central tendency of our estimator, $\mathbb{E}\left[h(X)\right]$, written

$$\mathrm{Bias}(h(X)) := r(X) - \mathbb{E}\left[h(X)\right],$$

where any non-zero difference between the pair is interpreted as a systematically positive or systematically negative error of the estimator, $h(X)$.

Variance measures the average deviation of a random variable from its expected value. In the current setting we are comparing the predicted value $h(X)$ of Y, with respect to some data $\mathcal{D}$ about the relationship between X and Y, and the average value of $h(X)$, $\mathbb{E}\left[h(X)\right]$, which we will write

$$\mathrm{Var}(h(X)) = \mathbb{E}\left[\left(\mathbb{E}\left[h(X)\right] - h(X)\right)^2\right].$$

The bias-variance decomposition of mean squared error is rooted in frequentist statistics, where the objective is to compute an estimate $h(X)$ of the true parameter $r(X)$ with respect to data $\mathcal{D}$ about the relationship between X and Y. Here the parameter $r(X)$ characterizing the truth about Y is assumed to be fixed and the data $\mathcal{D}$ is treated as a random quantity, which is exactly the reverse of Bayesian statistics. What this means is that the data set $\mathcal{D}$ is interpreted to be one among many possible data sets of the same dimension generated by the true model, the deterministic process $r(X)$.

Following (Bishop 2006), we may derive the bias-variance decomposition of mean squared error of h as follows. Let h refer to our estimate $h(X)$ of Y, r refer to the true value of Y, and $\mathbb{E}[h]$ the expected value of the estimate h. Then,

$$\begin{aligned}
\mathrm{MSE}(h) &= \mathbb{E}\left[(r-h)^2\right] \\
&= \mathbb{E}\left[((r-\mathbb{E}[h]) + (\mathbb{E}[h]-h))^2\right] \\
&= \mathbb{E}\left[(r-\mathbb{E}[h])^2\right] + \mathbb{E}\left((\mathbb{E}[h]-h)^2\right) + 2\mathbb{E}\left[(\mathbb{E}[h]-h)\cdot(r-\mathbb{E}[h])\right] \\
&= (r-\mathbb{E}[h])^2 + \mathbb{E}\left[(\mathbb{E}[h]-h)^2\right] + 0 \\
&= \mathrm{B}(h)^2 \ + \ \mathrm{Var}(h)
\end{aligned}$$

where the term $2\mathbb{E}\left[(\mathbb{E}[h]-h)\cdot(r-\mathbb{E}[h])\right]$ is zero, since

$$\begin{aligned}
\mathbb{E}\left[(\mathbb{E}[h]-h)\cdot(r-\mathbb{E}[h])\right] &= \left(\mathbb{E}[r\cdot\mathbb{E}[h]] - \mathbb{E}\left[\mathbb{E}[h]^2\right] - \mathbb{E}[h\cdot r] + \mathbb{E}[h\cdot\mathbb{E}[h]]\right) \\
&= r\cdot\mathbb{E}[h] - \mathbb{E}[h]^2 - r\cdot\mathbb{E}[h] + \mathbb{E}[h]^2 \qquad (5.5) \\
&= 0.
\end{aligned}$$

Note that the frequentist assumption that r is a deterministic process is necessary for the derivation to go through; for if r were a random quantity, the reduction of $\mathbb{E}[r\cdot\mathbb{E}[h]]$ to $r\cdot\mathbb{E}[h]$ in line (5.5) would be invalid.

One last detail that we have skipped over is the prediction error of $h(X)$ due to noise, N, which occurs independent of the model/learning algorithm used. Thus, the full bias-variance decomposition of the mean-squared error of an estimate h is the sum of the bias (squared), variance, and irreducible error:

$$\mathrm{MSE}(h) \ = \ \mathrm{B}(h)^2 \ + \ \mathrm{Var}(h) \ + \ N \qquad (5.6)$$

Chapter 6

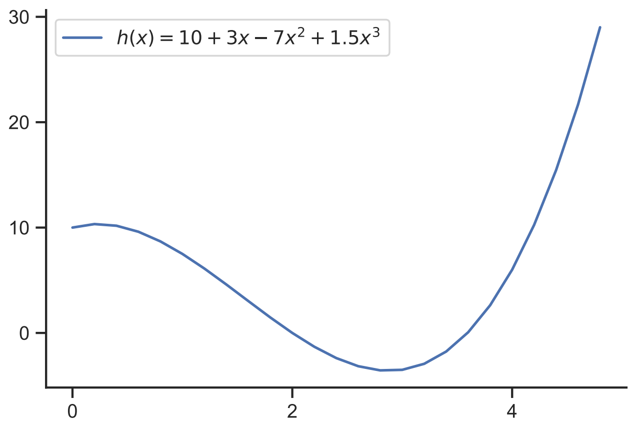
Decision Trees and Random Forests

We have focused on (proper) **linear models** and for good reason. They are computationally cheap, are readily interpretable, and typically easy to implement. In addition to proper linear models, improper linear models also perform surprisingly well for out-of-sample prediction on small data sets (Wheeler 2018, §4.3).

Linear models can capture non-linear responses, too. For example, consider the following nonlinear hypothesis

$$h = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \tag{6.1}$$

which we may plot for values of $0 \leq x \leq 5$ and vector $\boldsymbol{\theta} = [10, 3, -7, 1.5]$



However, by introducing the vector $z = [1, x, x^2, x^3]$ of **basis functions** of x, we may rewrite Equation 6.1 as

$$h = \theta_0 z_0 + \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3$$

which is linear in $\boldsymbol{\theta}$. Then, we may simply solve h by Equation 2.13, that is $h = \boldsymbol{\theta}^T \boldsymbol{z}$, for parameter vector $\boldsymbol{\theta}$ and feature vector $\boldsymbol{z}$ whose components are each polynomials of x.

Although **polynomial regression** is in the class of models linear in $\boldsymbol{\theta}$, applying polynomial regression to capture non-linearities depends on knowing the underlying basis functions. Unfortunately, those basis functions are typically unknown. Another issue is that, even when the right polynomial basis functions are known, the dimensionality of the original feature space increases. For large multi-feature data sets, therefore, polynomial regression usually is not a good option.

Rather than specify a particular basis function, another approach is to learn useful features $\phi(\boldsymbol{x})$ directly from input data $\boldsymbol{x}$. This approach is called an **adaptive basis model**, and **decision trees** are a popular type of adaptive basis model used to predict non-linear effects.

6.1 Classification and Regression Trees (CART)

Rather than built a model on every point in a data set, decision trees partition the data set recursively. For example, suppose our goal is to predict the value of a target variable Y from variables $X_1, X_2, \ldots, X_n$.

Classification and regression trees, or **CART** models, are nonparametric supervised learning models that employ a **decision tree** learning method. CART models are sometimes called decision trees,

Tree learning algorithms refer to a family of non-parametric, supervised learning techniques to learn a hierarchy—structured as a "tree"—of

6.2 Random Forests

6.3 Bagging and Boosting

Appendix A

Probability

This chapter reviews some basic facts about probability theory. For excellent textbooks on this subject, see (Jaynes 2003; Wasserman 2004; Bertsekas and Tsitsiklis 2008).

The building blocks of the standard, mathematical theory of probability due to Andrei Kolmogorov (Kolmogorov 1950) are an (i) *experiment*, which is any procedure that can be repeated, in theory, an infinite number of times; (ii) a *sample outcome*, which is any of the potential results from running an experiment; (iii) a *sample space*, which is the exhaustive set of sample outcomes of the experiment; and (iv) an *event*, which is a designated collection of sample outcomes, and are the objects to which a probability is assigned. These ingredients form the *measure theoretic* mathematical basis of probability axiomatized by Kolmogorov. Specifically, Kolmogorov's theory assumes there is a sample space Ω, a σ-algebra of events $\mathcal{A}$ over Ω, and a probability measure p over those events. To define conditional distributions, Kolmogorov uses Lebesque integrals to define expectation, then uses an integral equation based on the Radon-Nikodym theorem (Pollard 2002).

Probability is a rich subject, richer even than the standard textbook presentation we shall rehearse here. Although many view probability as nothing more than measure-theoretic probability some, notably Bruno de Finetti (1974) and Leonard Savage (1954), argued forcefully against the unreflective reliance on countable additivity and the machinery of σ-algebras. The mathematical convenience of σ-algebras ought not to be confused for a universal normative standard, anymore than the parallel postulate of Euclidean geometry ought to be confused for a universal fact about space.

It is humans who suffer uncertainty, not mathematics. Our mathematics then ought to serve our efforts to tame uncertainty, not the other way around.

A.1 Discrete Events and Probability

A *sample space* Ω is the set of possible outcomes of an experiment, elements ω in Ω are *outcomes*, and *events* are subsets of Ω. For example, a single coin toss is an experiment typically construed to have two possible outcomes, $\Omega = \{H, T\}$, with the sample outcome of heads (H) or tails (T), and events such as the coin landing heads $\{H\}$ or the event of the coin landing either heads or landing tails, $\{H \cup T\}$.

The *complement* of an event A, written A^c or sometimes written $\bar{A}$, is defined by the set of outcomes not in A, that is $A^c := \{\omega \in \Omega \mid \omega \notin A\}$. The *union* of events A and B is the set of outcomes either in A or in B, that is $A \cup B := \{\omega \in \Omega \mid \omega \in A \text{ or } \omega \in B\}$. The *intersection* of events A and B is the set of outcomes that are both in A and in B, that is $A \cap B := \{\omega \in \Omega \mid \omega \in A \text{ and } \omega \in B\}$. The sets $A_1, A_2, \ldots$ are *disjoint* if $A_i \cap A_j = \emptyset$, for $i \neq j$. Finally, the *set difference* between A and B, defined by $A - B := \{\omega \in \Omega \mid \omega \in A \text{ and } \omega \notin B\}$ captures the (strict) subset relation, $A \subset B$.

Probability Axioms

Definition A.1. A *probability mass function* p is a real-valued function defined on an algebra $\mathcal{A}$ over a set of states Ω satisfying the following three conditions:

(P1) **(Non-negativity)** $p(A) \geq 0$ for every $A \in \mathcal{A}$;

(P2) **(Normalization)**$p(\Omega) = 1$;

(P3) **(Additivity)** If A_1 and A_2 are disjoint events in $\mathcal{A}$, then

$$p(A_1 \cup A_2) = p(A_1) + p(A_2)$$

More generally, if Ω has a countably infinite number of members and $A_1, A_2, \ldots$ are disjoint events of $\mathcal{A}$ over Ω, then

$$p\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} p(A_i).$$

In plain terms, p is a single probability function assigning to each *event* in the algebra $\mathcal{A}$ a numerically determinate real number. The triple $(\Omega, \mathcal{A}, p)$ is called a *probability space*.

For arbitrary events A and B in $\mathcal{A}$, an immediate consequence of properties (P1) – (P3) is a generalization of (P3):

(P3*) $p(A \cup B) = p(A) + p(B) - p(A \cap B)$.

Several natural properties of probability follow from these axioms. For instance, from normalization and additivity we may derive that the probability of the empty event is 0:

$$\begin{aligned} 1 &= p(\Omega) \\ &= p(\Omega \cup \emptyset) = p(\Omega) + p(\emptyset) \\ &= 1 + p(\emptyset) \end{aligned}$$

Conditional Probability

Definition A.2. *Conditional probability* of A given B, written $p(A \mid B)$, is defined as a ratio of probabilities:

$$p(A \mid B) := \frac{p(A \cap B)}{p(B)}, \quad \text{when } p(B) > 0$$

Observe that, for any given B such that $p(B) > 0$, the function $p(\cdot \mid B)$ satisfies the probability axioms for events plugged into the left side of the bar.

The *product rule* defines the probability of the joint event of A and B in terms of a conditional probability and the marginal probability of the conditioning event, namely

$$p(A, B) := p(A \cap B) = p(A \mid B)p(B) = p(B \mid A)p(A).$$

So far we have been considering binary events X, such that $X = 1$ or $X = 0$, representing the state where X is true or the state where X is false, respectively. We may generalize by defining a *discrete random variable* X as a real-valued function that maps experimental outcomes to real numbers. .

Discrete Random Variable

Definition A.3. A **random variable** X is a real-valued function from a finite or countably infinite Ω to $\mathbb{R}$ that assigns a real number $X(\omega)$ to each outcome $w \in \Omega$.

Machine learning and statistics are primarily concerned with data, and random variables link sample spaces and events to data. In fact, most applications of probability work directly with random variables and do not mention sample spaces. This is sometimes reflected in the notation used to describe the event $X = x$ of the random variable taking a particular value. We too will follow this notation. However, bear in mind that sample spaces remain present in the backgrounds.

We may express the probability of the event X taking value x as $p(X = x)$ or $p(x)$ for short. Binary events then are special cases of discrete random variables. Specifically, a binary *indicator function* $\mathbf{1}$ of a subset A of Ω is a map from Ω to 0 or 1, defined by

$$\mathbf{1}_A(x) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A \end{cases}$$

Given a joint probability $p(A, B)$, the *marginal distribution* of A is defined by

$$p(A) = \sum_b p(A \mid B = b)p(B = b)$$

where the summation is over all states of B. This summation is sometimes referred to as the *rule of total probability*.

Combining the definition of conditional probability with the product rule and total probability rule yields *Bayes' Theorem*.

Bayes' Theorem

Theorem A.1. For discrete random variables X and Y with particular events $X = x$ and $Y = y$,

$$p(X = x \mid Y = y) = \frac{p(Y = y \mid X = x)p(X = x)}{\sum_{x^* \in \mathcal{X}} p(Y = y \mid X = x^*)p(X = x^*)}$$

where
$p(X = x \mid Y = y)$ is called the **posterior distribution**,
$p(Y = y \mid X = x)$ is called the **likelihood**, and
$p(X = x)$ is called the **prior distribution**.

Two events A and B are *probabilistically independent*, written $A \perp\!\!\!\perp B$, when their joint probability is equal to the product of the marginal probabilities. In symbols,

$$A \perp\!\!\!\perp B := p(A, B) = p(A)p(B)$$

Equivalently, we say the pair is independent when the marginal of A is equal to the conditional probability of A given B, and vice versa. More generally, we say that A and B are *conditionally independent* of C just in case the conditional joint distribution is a product of the conditional marginal distributions. In symbols,

$$A \perp\!\!\!\perp B \mid C := p(A, B \mid C) = p(A \mid C)p(B \mid C)$$

An important point to keep in mind is that while probabilistic independence refers to events, it is a property of a probability distribution defined over those events and their intersection.

A.2 Odds Ratios and Bayes Factor

The *odds ratio* of an event A is the probability of A occurring divided by the probability of A not occurring:

$$odds(A) := \frac{p(A)}{p(\bar{A})} = \frac{p(A)}{1 - p(A)}$$

Alternatively, we may calculate the probability of an event A in terms of odds ratios:

$$p(A) := \frac{odds(A)}{(1 + odds(A))}$$

The *Bayes factor* BF is the ratio of two conditional probabilities,

$$BF = \frac{p(D \mid A)}{p(D \mid \bar{A})}$$

where $p(D|A)$ is understood to be the probability of receiving data D given the occurrence of the event A, and $p(D \mid \bar{A})$ is the probability of receiving data D given the occurrence of the complement of A. If the Bayes factor is greater than 1, then the data has increased our belief (i.e., increased our personal probability p) in the event A occurring. Similarly, if the Bayes factor is less than 1, then the data makes us think the event is less probable than we originally believed.

The Bayes factor encodes the impact data D has on our estimate of A. More specifically, it is the factor by which *prior odds* of A is changed to the *posterior odds* of A occurring:

$$prior\ odds(A) \cdot BF = posterior\ odds(A)$$

A.3 Measures of Location

We often wish to summarize data numerically, and one important aspect of a data set distribution is its location on the number line. The most common estimator of the *central tendency* of a dataset is its *mean*. Turning to probabilities, a probability mass function assigns probabilities to all possible values of a random variable X and, similarly, you might wish to summarize this information by a single number. The *expectation* of a random variable X is the weighted average probability values of X.

Expectation

Definition A.4. The mean or **expectation** of a random variable X with respect to a probability mass function p is

$$\mathbb{E}[X] = \mu = \sum_{x \in X} xp(x)$$

To illustrate, suppose that a three-sided die (e.g., a can) is tossed, with sides labeled 1, 2, and 3. Suppose the probability of landing on side 1 is $^1/_6$, the probability landing on side 2 is $^1/_3$, and the probability of it landing on side 3 is $^1/_2$. The expected value of a single toss of this die is $1^1/_6 + 2^1/_3 + 3^1/_2 = \frac{7}{3}$.

A.4 Measures of Spread

Another important quantity that we may use to describe a data set distribution is the *spread* or dispersion. Intuitively, the measures of dispersion tell us something about the range of values that the data take. Turning to probabilities, we may define the *second moment* of the random variable X, with respect to a probability mass function p, as the expected value of X^2. Using this terminology, we can define the nth moment of a random variable as $\mathbb{E}[X^n]$. Hence, the mean of a random variable is simply the first moment of X.

The *variance* of a random variable X is the most common quantity measuring the dispersion of X around its mean.

Variance

Definition A.5. The **variance** of a random variable X with respect to a probability mass function p is

$$\mathrm{var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

which is the expected value of the dispersion of X around its mean. Using the **expected value rule** for functions of random variables, the variance can be calculated by

$$\mathrm{var}(X) = \sum_x (x - \mu)^2 \, p(x).$$

The **standard devision** of X is $\sigma_X = \sqrt{\mathrm{var}(X)}$.

A.5 Continuous Random Variables

Our review of probability thus far has focused exclusively on discrete quantities. In this section we consider how to extend probabilities to continuous quantities.

Suppose X is an uncertain, continuous quantity whose probability lies within the interval $[a, b]$. X is simply a *continuous random variable* which may take all possible real numbers between a and b. Since there are an uncountably infinite number of real numbers in any interval, so the probability of X taking any particular value in this span is zero. The upshot is that the methods used for constructing a discrete probability mass function do not apply to continuous random variables.

Probability Density Function

Definition A.6. A random variable X is **continuous** if there is a non-negative, normalized real-valued function f_X, called the **probability density function** (pdf) of X such that

$$p(X \in A) = \int_A f_X(x)dx$$

for every subset A of the real line.[a] The probability that X falls within the interval $[a, b]$ is

$$p(a \leq X \leq b) = \int_a^b f_X(x)dx$$

which may be interpreted as the area under the graph of the pdf. Figure A.1 illustrates one example.

[a] We assume this integral is well-defined. For exceptions, see (Seidenfeld, Schervish, and Kadane 2001). However, if f_X is a piecewise continuous function with finite our countably many discontinuous points, and A is a union of a finite or countable number of intervals, then technical complications and impossibility results are avoided.

To make the properties of non-negativity and normalization explicit, suppose X is a continuous random variable with pdf f_X. Then, f_X satisfies

(Non-negativity) $f_X(x) \geq 0$, for all $x \in X$

(Normalization) $\int_{-\infty}^{+\infty} f_X(x)dx = 1.$

The *expected value* of a continuous random variable X, written $\mathbb{E}(X)$ or μ, is

$$\mathbb{E}(X) = \mu = \int_{-\infty}^{+\infty} x \cdot f_X(x)dx$$

The *variance* of a continuous random variable X is

$$\text{var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \int_{-\infty}^{+\infty} (x-\mu)^2 \cdot f_X(x)dx$$

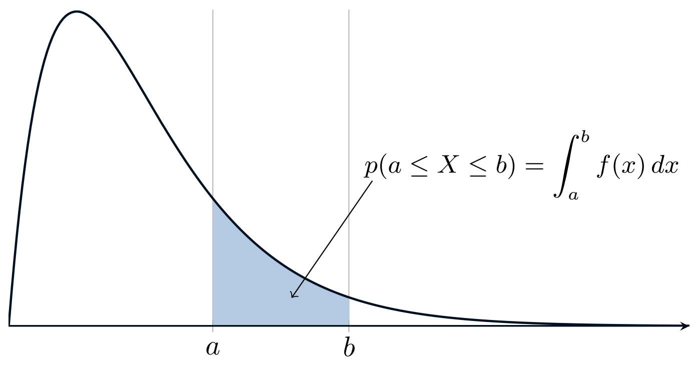


Figure A.1: Probability density of an event A.

A.6 Some Common Discrete Distributions

Suppose we flip a coin once with a probability θ of landing heads. Let X be a discrete random variable taking values from $\{0, 1\}$, where "success" or a **positive** outcome is $X = 1$ and "failure" or a **negative** outcome is $X = 0$. (By convention, the outcome "heads" is typically identified with success.) If the probability of success is θ, that is $P(X = 1) = \theta$, then the **Bernoulli distribution**, characterized by the probability distribution

$$P(X = n) = \theta^n(1-\theta)^{1-n} \quad \text{for } n \in \{0, 1\}$$

Now suppose we flip a coin n times with a probability θ of landing heads. Let X be a discrete random variable taking values from the set $\mathcal{X} = \{1, 2, \ldots, n\}$, representing the number of times the coin lands heads. Then, X has a *binomial distribution.*

Binomial Distribution

Theorem A.2. Suppose there are n independent trials, each resulting in one of two outcomes: *success* or *failure*. Let $\theta \in [0, 1]$ represent the probability that success occurs at any given trial, and assume that θ is stationary (i.e., the same) across all trials. Let X denote the total number of successes across n trials. Then, X has a **binomial distribution** and the probability of k heads in n trials is

$$p(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad \text{for } 0, 1, \ldots, n,$$

where $\binom{n}{k}$ is $\frac{n!}{k!(n-k)!}$.

The **mean** of X is $\mathbb{E}[X] = np$ and the **variance** is $\text{var}(X) = np(1-p)$.

We can see that the Bernoulli distribution is a special case of the binomial distribution, namely when $n = 1$.

Urn models represent experiments whose events resemble drawing samples without replacement from an urn containing two different colored balls.

Hypergeometric Distribution

Theorem A.3. Suppose there is an urn containing r red balls and w white balls such that $r + w = N$. If n balls are drawn at random without replacement, and X denotes the total number of red balls selected, then X has a **hypergeometric distribution** and the probability that $X = k$ is

$$p(X = k) = \frac{\binom{r}{k}\binom{w}{n-k}}{\binom{N}{n}},$$

where k varies over all integers for which both $\binom{r}{k}$ and $\binom{w}{n-k}$ are defined.

The **mean** of X is $\mathbb{E}[X] = \frac{rn}{N}$ and the **variance** is $\text{var}(X) = \frac{nr(N-r)(N-n)}{N^2(N-1)}$.

Suppose we have a data set consisting of m elements. In statistical parlance, we would describe this data as a population of size m whose members have the values $\{x_1, \ldots, x_m\}$ of some particular measurement, such as each members weight in kilograms. The value of that measurement of a randomly drawn individual from the population — the weight in kilograms of a randomly selected member from our population of m individuals — has a probability distribution which is called the *empirical distribution*.

Empirical Distribution

Definition A.7. Suppose $\mathcal{D} = \{x_1, \ldots, x_m\}$ is a set of data. The **empirical distribution** associated with $\mathcal{D}$ is the expectation of the value of some function $g(\cdot)$ over the data, defined by

$$\mathbb{E}_m\left[g(X)\right] := \frac{1}{m}\sum_{i=1}^{m} g(x_i)$$

When $g(x) = x$, the **mean** of the empirical distribution, written $\bar{x}_m$, is

$$\mathbb{E}_m\left[g(X)\right] := \bar{x}_m = \frac{1}{m}\sum_{i=1}^{m} x_i$$

The **variance** of the empirical distribution is

$$\begin{aligned}\mathrm{var}_m(X) &= \mathbb{E}_m\left[(X - \mathbb{E}_m\left[X\right])^2\right] \\ &= \mathbb{E}_m\left[(X - \bar{x}_m)^2\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x}_m)^2\end{aligned}$$

A.7 Some Common Continuous Distributions

The most commonly used distribution in statistics and machine learning is the *Gaussian* distribution, which is also called the *normal distribution.*

Gaussian (Normal) Distribution

Definition A.8. continuous random variable X is **Gaussian** or **normal** if it has a probability density function f_X of the form

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

where μ and σ are scalar parameters characterizing the pdf, with $\sigma > 0$.

The **mean** (and mode) is $\mu = \mathbb{E}[X]$, and the **variance** is $\sigma^2 = \mathrm{var}(X)$.

We write $X \sim \mathcal{N}(\mu, \sigma^2)$ to denote that X follows a normal distribution with mean μ and variance σ^2. If $X \sim \mathcal{N}(0, 1)$ we say that X follows a **standard normal distribution**.

The **precision** of a Gaussian is the inverse of its variance, $1/\sigma^2$. Thus, a high precision Gaussian pdf refers to a high-peaked distribution with low variance centered on μ.

There are several reasons why Gaussian distributions are used.

1. The two parameters, μ and σ^2, are both easy to interpret and capture basic properties of a distribution: location and dispersion.

2. The central limit theorem says that the sums of independent random variables have approximately Gaussian distributions, which is a strong justification for using the Gaussian distribution to model residual errors or "noise" in i.i.d. experiments.

3. With only two necessary parameters, mean and variance, the Gaussian distribution is conservative in the number of assumptions it makes about data and thus is least biased in the sense of maximizing entropy when no other information about a distribution is available.

4. The distribution is easy to implement yet highly effective in a wide range of cases.

Nevertheless, one problem with the Gaussian distribution is that it is sensitive to outliers. A distribution that is more robust against outliers is *Student's* t *distribution.*

t Distribution

Definition A.9. For continuous random variable X,

$$\tau(X) \propto \left[1 + \frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right]^{-\left(\frac{\nu+1}{2}\right)}$$

where μ is the **mean** (and mode), $0 < \sigma^2$ is the scale parameter, and $0 < \nu$ is the **degrees of freedom**. The **variance** is

$$\frac{\nu\sigma^2}{(\nu-2)}$$

The *Beta* distribution is another important family of distributions, particularly for Bayesian inference.

Beta Distribution

Definition A.10. For continuous random variable X that takes values between 0 and 1, the pdf is

$$\mathrm{B}(x \mid a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$$

where ${}^{\Gamma(a+b)}/_{\Gamma(a)\Gamma(b)}$ is a constant to ensure the curve with shape $x^{a-1}(1-x)^{b-1}$, which is called the **beta function** and is denoted by $beta(a, b)$, is a probability density function.

$\Gamma(c)$ is the **gamma function**, which is defined by $\Gamma(c) = (c-1)!$ when c is an integer, otherwise by

$$\Gamma(x) := \int_0^\infty u^{x-1} e^{-u} du$$

The **mean** of a beta distribution is

$$\mathbb{E}[X] = \frac{a}{a+b}$$

The **variance** of a beta distribution is

$$\mathrm{var}(X) = \mathbb{E}[X^2] - [\mathbb{E}[X]]^2 = \frac{ab}{(a+b)^2(a+b+1)}$$

The **uniform distribution** is the special cases of a beta distribution when $a = b = 1$, that is $\mathrm{beta}(1, 1)$.

Appendix B

Information Theory

Suppose you want to communicate a message from point A to point B through some communication channel C. By how much can you compress the message without losing information through the channel? (The answer is the entropy, H). What is the upper limit on the transmission rate between A and B? (The answer is the capacity of the channel, C).

Information theory provides a powerful framework for answering these two fundamental questions, and it turns out that it applies to a wide range of other fields. Cover and Thomas's textbook, *Elements of Information Theory* (2006), is a classic.

B.1 Entropy

Entropy is a quantity defined in terms of a probability distribution which measures the uncertainty of a random variable.

Entropy

Definition B.1. Suppose X is a discrete random variable and p a probability mass function. We write $p(x)$ to denote the event $p(X = x)$, where $x \in \mathcal{X}$. Then **entropy** of X, written $H(X)$, is defined by

$$H(X) = -\sum_{x\in\mathcal{X}} p(x) \log p(x)$$

Properties:
1. $H(X) \geq 0$
2. $H_b(X) = (\log_b a) H_a(X)$

When the base of the logarithm is 2, entropy is measured in **bits**. When the base of the logarithm is e, the unit of measure of entropy is **nats**. Unless specified otherwise, the unit of measure in information theory is bits, so the logarithms are assumed to have base 2.

Joint and Conditional Entropy

Definition B.2. The **joint entropy** of a pair of discrete random variable (X, Y), written $H(X, Y)$, is defined with respect to a joint probability distribution $p(X, Y)$ by

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y)$$

The **conditional entropy** $H(Y \mid X)$ for a pair of discrete random variables (X, Y) parameterized by a joint probability distribution $p(X, Y)$ is

$$H(Y \mid X) = \sum_{x\in mathscrX} p(x)H(Y \mid X = x)$$

Some common properties of entropy are

1. **(Non-negativity)** $H(X) \geq 0$
2. **(Rescaling)** $H_b(X) = (\log_b a)H_a(X)$
3. **(Chain Rule)** $H(X, Y) = H(X)H(Y \mid X)$
4. **(Conditioning Reduces Entropy)** For any two random variables X, Y,

$$H(X \mid Y) \leq H(X)$$

   and $H(X \mid Y) = H(X)$ iff $X \perp\!\!\!\perp Y$.
5. $H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$, but generally $H(Y \mid X) \neq H(X \mid Y)$.

The entropy of a random variable is a measure of the uncertainty of the random variable. It measures the amount of information, on average, necessary to describe the random variable. Relative entropy affords a comparison of two distributions in terms of the relative inefficiency of operating with a distribution q *vis a vis* the true distribution p. Informally, if the true distribution p was known for a random variable, we the average description length of a message that we could construct, measured in bits, is $H(p)$. If instead we used another distribution q for the same task, we would need $H(p)$ bits plus the difference in bits between using p and using q, which is written $D(p\|q)$. This difference is the relative entropy between p and q, which is also called the *Kullback-Leibler divergence*.

Kullback-Leibler Divergence

Definition B.3. The **Kullback-Leibler** or the **relative entropy** between two probability mass functions $p(x)$ and $p(x)$, with respect to a discrete random variable X, is

$$\begin{aligned} D(p\|q) &= \sum_{x \in \mathcal{X}} \log \frac{p(x)}{p(x)} \\ &= \mathbb{E}\left[\log \frac{p(X)}{q(X)}\right] \end{aligned}$$

Corner cases in this definition are handled by adopting the following conventions: we stipulate that

- $0 \log \frac{0}{0} = 0$
- $0 \log \frac{0}{q} = 0$
- $p \log \frac{p}{0} = \infty$

Relative entropy is non-negative and is zero when $p = q$. Relative entropy is not a distance measure because it does not satisfy the triangle inequality. In general, $D(p\|q) \neq D(q\|p)$. Hence the name KL-divergence rather than KL-“distance”.

We might ask another question of the relationship between random variables, which is the amount of information about one random variable that is contained in another. In information theoretic terms this question asks what, if any, amount of uncertainty about one random variable is reduced by knowledge of another. This quantity is called *mutual information.*

Mutual Information

Definition B.4. Let X, Y be a pair of discrete random variables with a joint probability mass function $p(x, y)$ and marginal probabilities $p(x)$ and $p(y)$. The **mutual information** of X and Y, written $I(X; Y)$, is the relative entropy between the joint distribution and the product of the marginals:

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{X}} \sum_{x \in \mathcal{X}} \log \frac{p(x,y)}{p(x)p(y)} \\ &= D\left(p(x,y) \| p(x)p(y)\right) \\ &= \mathbb{E}\left[\log \frac{p(X,Y)}{p(X)p(Y)}\right] \end{aligned}$$

Equivalently, we may define mutual information in terms of entropy:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X).$$

Appendix C

Linear Algebra

Linear algebra provides a set of conventions for representing and operating on a set of linear equations. An excellent textbook is (Gentle 2007). Here we present a very brief review.

Many machine learning algorithms involve matrix and vector arithmetic. For example, an advanced optimization algorithm to find the minimum of a loss function may use a vector of first-derivative terms together with a matrix of second derivative terms.

To start we may think of matrices as rectangular arrays of scalars (i.e., as elements of a field). We shall assume throughout that those scalar values are real numbers.

For example, we might have the following system of two linear equations:

$$\begin{aligned} 7x_1 - 4x_2 &= 9 \\ -2x_1 + 3x_2 &= 0 \end{aligned}$$

Sometimes systems of equations admit multiple solutions, such as when one equation is a function of another. But, for many cases, like the one above, there is a unique solution. Here is is $x_1 = 3; x_2 = 2$.

Matrix notion allows us to give a compact representation of this system as

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 7 & -4 \\ -2 & 3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 9 \\ 0 \end{bmatrix}$$

In general, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ refers to a matrix with m rows and n columns. In our example above, $\boldsymbol{A}$ is in $\mathbb{R}^{2\times 2}$. By convention, an n-dimensional vector is a *column vector* with a single column and n rows.

C.1 Vectors

The number n of elements in an n-dimensional vector is the **order** of a vector. An n-dimensional vector may be thought of as representing a point in n-dimensional space.

We may write a vector $\boldsymbol{x}$ and its *transpose*, $\boldsymbol{x}^T$, as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$$

By convention, a vector $\boldsymbol{x}$ is a single column and the transpose of $\boldsymbol{x}$ "flips" that column to a single row. Alternatively, you sometimes see the terms *column vector* and *row vector* used. We will give a general definition of the transpose operation for matrices, later.

An important operation on two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same order is the

The **inner product** of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same order is called the **inner product**, which is also called the **dot product**. As a reminder once more, we are restricting our attention to real vectors.

Inner Product

Definition C.1. Given two vectors of the same order, written $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, the quantity $\boldsymbol{x}^T\boldsymbol{y}$ is the **inner product** of the vectors, which is a single real number computed by

$$\boldsymbol{x}^T\boldsymbol{y} \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i$$

The inner product is a special case of matrix multiplication, which we will see below.

Given two vectors not necessarily of the same order, $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x}\boldsymbol{y}^T$ defines the **outer product** of the vectors, which is a matrix whose elements are calculated by

$$\boldsymbol{x}\boldsymbol{y}^T \in \mathbb{R}^{m\times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \times \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_n \\ x_2y_1 & x_2y_2 & \cdots & x_2y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_my_1 & x_my_2 & \cdots & x_my_n \end{bmatrix}$$

Informally, a **norm** of a vector is a measure of its length. For example, the **Euclidean** or the $\boldsymbol{\ell_2}$**-norm** (pronounced "L2") is

$$\|\boldsymbol{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$$

Many norms can be defined for vectors. The Euclidean norm belongs to an important class of norms called the ℓ_p-**norms**, which are defined by

$$\|\boldsymbol{x}\|_p := \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

when $p \geq 1$. An ℓ_p norm is also called a p-norm, the *Minkowski norm*, and also the *Hölder norm*.

When $p = 1$, $\|\boldsymbol{x}\|_1 := \sum_i |x_i|$. The ℓ_1-norm is called the *Manhattan norm* because it measures distance by summing along coordinate axes, like one might travel along the grid pattern of city blocks that (most of) the streets in Manhattan are laid out on.

When $p = \infty$, $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$.

Norms

Definition C.2. A **norm** is a function f from an n-dimension vector to a real number, $\mathbb{R}^n \mapsto \mathbb{R}$, satisfying the following four properties

1. (**Non-negativity**) For all vectors $\boldsymbol{x} \in \mathbb{R}^n$, then $f(\boldsymbol{x}) \geq 0$
2. (**Additive Identity**) $f(0) = 0$
3. (**Homogeneity of Scalar and Real Multiplication**) For all $\boldsymbol{x} \in \mathbb{R}^n$ and $a \in \mathbb{R}$, $f(a\boldsymbol{x}) = af(\boldsymbol{x})$
4. (**Triangle Inequality**) For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$

Instead of writing $f(\boldsymbol{x})$ to express the norm of $\boldsymbol{x}$, we instead write $\|\boldsymbol{x}\|$.

There is a close relationship between norms and inner products. Consider a vector $\boldsymbol{x}$. A norm of $\boldsymbol{x}$ may be defined in terms of the square root of the inner product of $\boldsymbol{x}$ and itself:

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

This is called the *norm induced by an inner product.*

C.2 Matrices

An n dimensional vector is a $1 \times n$ dimensional matrix. It is common to think of a vector and its transpose as matrices. Indeed, while we gave an informal definition of vectors and their transposes, here we give the formal definition of the transpose operation in terms of matrices.

Transpose

Definition C.3. For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ the **transpose** of $\boldsymbol{A}$, written $\boldsymbol{A}^T$, is the $n \times m$ matrix defined by

$$(\boldsymbol{A}^T)_{i,j} = \boldsymbol{A}_{j,i}$$

where the element in the jth column and ith row of $\boldsymbol{A}$ is assigned to the ith row and jth column.

Properties of transposes:
- $(\boldsymbol{A}^T)^T = \boldsymbol{A}$
- $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$
- $(\boldsymbol{A}+\boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$

Yet one may also view the rows and columns of a matrix as vectors. The columns of an $m \times n$ matrix generates a vector space of order n and of dimension m (or less), and is called the *column space* or *manifold* of $\boldsymbol{A}$.

There are a variety of multiplication operations one can perform on matrices. The most common type is called *Cayley multiplication*, but is often simply referred to as **matrix multiplication**.

Matrix Multiplication

Definition C.4. Suppose $\boldsymbol{A}$ is an $n \times m$ matrix and $\boldsymbol{B}$ is a $m \times q$ matrix. **Cayley** or **matrix multiplication** of $\boldsymbol{A}$ and $\boldsymbol{B}$ is a mapping from $\mathbb{R}^{n\times m} \times \mathbb{R}^{m\times q} \mapsto \mathbb{R}^{n\times q}$, where $\boldsymbol{C}$ is a $n \times q$ matrix with elements

$$c_{i,j} = \sum_k a_{i,k} b_{k,j}.$$

Matrix multiplication is *not* commutative: generally, $\boldsymbol{AB} \neq \boldsymbol{BA}$, even for square matrices. However, matrix multiplication is

1. **(Associative)** $\boldsymbol{A}(\boldsymbol{BC}) = (\boldsymbol{AB})\boldsymbol{C}$
2. **(Distributive)** $\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB} + \boldsymbol{AC}$ and $(\boldsymbol{B}+\boldsymbol{C})\boldsymbol{A} = \boldsymbol{BA} + \boldsymbol{CA}$

One can view matrix multiplication as a set of vector-vector products. For instance, the (i, j)th element of $\boldsymbol{C}$ is equal to the inner produce of the ith row of $\boldsymbol{A}$ and the jth row of $\boldsymbol{B}$. Schematically, we have

$$\boldsymbol{AB} = \boldsymbol{C} = \begin{bmatrix} - & \boldsymbol{a}_1^T & - \\ - & \boldsymbol{a}_2^T & - \\ & \vdots & \\ - & \boldsymbol{a}_m^T & - \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_q \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^T\boldsymbol{b}_1 & \boldsymbol{a}_1^T\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_1^T\boldsymbol{b}_q \\ \boldsymbol{a}_2^T\boldsymbol{b}_1 & \boldsymbol{a}_2^T\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_2^T\boldsymbol{b}_q \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_m^T\boldsymbol{b}_1 & \boldsymbol{a}_m^T\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_m^T\boldsymbol{b}_q \end{bmatrix}$$

The **identity matrix**, denoted $\boldsymbol{I}_n \in R^{n \times b}$, is a square matrix with 1s on the diagonal and zeros everywhere else. For example,

$$\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is the 3-by-3 identity matrix. Formally, for elements $i, j \in I$, $\boldsymbol{I}$ is an identity matrix just in case

$$\boldsymbol{I}_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

In general, the index n is dropped and the dimension of the identity matrix is inferred by context. With this relaxation of notation, we can say of non-square matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ that

$$\boldsymbol{A}\boldsymbol{I} = \boldsymbol{A} = \boldsymbol{I}\boldsymbol{A}$$

bearing in mind that $\boldsymbol{I}$ appearing in $\boldsymbol{A}\boldsymbol{I}$ is a $n \times n$ identity matrix, while $\boldsymbol{I}$ appearing in $\boldsymbol{I}\boldsymbol{A}$ is an $m \times m$ identity matrix. These dimension are determined by which will support matrix multiplication.

Inverse

Definition C.5. The **inverse** of a square matrix $\boldsymbol{A}^{n \times n}$, written $\boldsymbol{A}^{-1}$, is the unique matrix (if defined) such that

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = I_n$$

When $\boldsymbol{A}^{-}1$ of $\boldsymbol{A}$ exists, the follow properties hold:

- $(\boldsymbol{A}^{-1})^{-1} = \boldsymbol{A}$
- $(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$

Non-square matrices do not have inverses, by definition. However, some square matrices do not have inverses, either. We say that a matrix $\boldsymbol{A}$ is **invertible** or **non-singular** if $\boldsymbol{A}^{-1}$ exists and **non-invertible** or **singular** otherwise.

The relationship between the inverse and transpose of a matrix is

$$(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1}$$

Given this identity, the **inverse of a transpose** of a matrix is often written $\boldsymbol{A}^{-T}$.

The **trace** and **determinant** are two common, scalar-valued operations on square matrices. That is, each is a map from $\mathbb{R}^{n \times n}$ to $\mathbb{R}$. Although each definition is straightforward, they each appear in many applications of matrix algebra.

Trace

Definition C.6. The **trace** of a square matrix $\boldsymbol{A}^{n\times n}$, written $\operatorname{tr}(\boldsymbol{A})$, is the sum of the diagonal elements of $\boldsymbol{A}$:

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{i,i}$$

When $\boldsymbol{A}$ and $\boldsymbol{B}$ are square matrices, the follow properties hold:

- $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{A}^T)$
- $\operatorname{tr}(\boldsymbol{A}+\boldsymbol{B}) = \operatorname{tr}(\boldsymbol{A}) + \operatorname{tr}(\boldsymbol{B})$
- $\operatorname{tr}(a\boldsymbol{A}) = a(\operatorname{tr}(\boldsymbol{A}))$, for $a \in \mathbb{R}$.

Integers are often used as labels for things. Phone numbers and football player jersey numbers use integers as names. We don't add phone numbers, nor order players by their number, even if there are some conventions for how those numbers are assigned. When we use numbers merely as labels we are sometimes are interested in how many different ways those objects can be reordered. The set of labels $\{1, 2, 3\}$, for example, can be rearranged $3! = 6$ ways, or $3 \times 2 \times 1$. In general, there are $n!$ **permutations** of the integers 1 to n. To refer to the jth permutation within the set S of $n!$ permutations, we use the notation $\sigma_j = (j_1, \dots j_n)$. The **determinant** of a matrix is an operation that multiplies different permutations of elements of a matrix.

More specifically, suppose $\boldsymbol{A}$ is an $n \times n$ square matrix and

$$\prod a_{i,\sigma_j} = a_{1,j_1} \times \cdots \times a_{n,j_n}$$

is the product of the k-th permutation of the index labels 1 to n. We say that a permutation is **odd** or **even** according to the number of times a smaller element follows a larger one in a given permutation. The **signature** of a permutation is its parity, defined by

$$\operatorname{sgn}(\sigma) = \begin{cases} -1 & \text{if } \sigma \text{ is an even permutation} \\ 1 & \text{otherwise} \end{cases}$$

For example, given the set of labels $\{1, 2, 3\}$, the signatures of the tuples $(1, 2, 3)$ and $(3, 2, 1)$ are both even, and the signature of the tuples $(1, 3, 2)$ and $(2, 1, 3)$ and $(1, 3, 2)$ are both odd.

Determinant

Definition C.7. Let $\boldsymbol{A}$ be an $n \times n$ square matrix and a_{k,j_i} be element a_k of $\boldsymbol{A}$ appearing in the i-th position with in the j-th permutation of 1 to n. The **determinant** of $\boldsymbol{A}$, written $\det(\boldsymbol{A})$, is defined by

$$\det(\boldsymbol{A}) = \sum_{\sigma_j \in S} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma_j} \right)$$

where $\operatorname{sgn}(\sigma)$ is either 1 or -1.

C.3 Vector Calculus

Suppose $\boldsymbol{v} = [v_1, v_2, \ldots, v_n]$ is an n-dimensional vector and x is a scalar. Observe that differentiating a vector $\boldsymbol{v}$ by a scalar x is a vector of partial derivatives, written

$$\frac{\partial \boldsymbol{v}}{\partial x} = \left[\frac{\partial v_1}{\partial x}, \frac{\partial v_2}{\partial x} \ldots, \frac{\partial v_n}{\partial x}\right]$$

and differentiating a scalar x by the differentiating vector $\boldsymbol{v}$ is a vector of the form

$$\frac{\partial x}{\partial \boldsymbol{v}} = \left[\frac{\partial x}{\partial v_1}, \frac{\partial x}{\partial v_2} \ldots, \frac{\partial x}{\partial v_n}\right]$$

Now suppose $\boldsymbol{t} = [t_1, t_2, \ldots, t_m]$ is an m-dimensional vector. Then, differentiating vector $\boldsymbol{t}$ by vector $\boldsymbol{v}$ yields an $m \times n$ matrix of the form

$$\frac{\partial \boldsymbol{t}}{\partial \boldsymbol{v}} = \begin{bmatrix} \frac{\partial t_1}{\partial v_1} & \frac{\partial t_1}{\partial v_2} & \cdots & \frac{\partial t_1}{\partial v_n} \\ \frac{\partial t_2}{\partial v_1} & \frac{\partial t_2}{\partial v_2} & \cdots & \frac{\partial t_2}{\partial v_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial t_m}{\partial v_1} & \frac{\partial t_m}{\partial v_2} & \cdots & \frac{\partial t_m}{\partial v_n} \end{bmatrix}$$

C.4 Some Useful Results

We collect here some results from matrix algebra. Let $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ be matrices and $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors. Then, we report the following equations:

$$\frac{\partial(\boldsymbol{a}^T \boldsymbol{b})}{\partial \boldsymbol{b}} = \boldsymbol{a} \tag{C.1}$$

$$\frac{\partial(\boldsymbol{a}^T \boldsymbol{A} \boldsymbol{a})}{\partial \boldsymbol{a}} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{a} \tag{C.2}$$

$$\frac{\partial}{\partial \boldsymbol{A}} \mathrm{tr}(\boldsymbol{B}\boldsymbol{A}) = \boldsymbol{B}^T \tag{C.3}$$

$$\frac{\partial}{\partial \boldsymbol{A}} \log(\det \boldsymbol{A}) = (\boldsymbol{A}^{-1})^T =: \boldsymbol{A}^{-T} \tag{C.4}$$

$$\mathrm{tr}(\boldsymbol{ABC}) = \mathrm{tr}(\boldsymbol{CAB}) = \mathrm{tr}(\boldsymbol{BCA}) \tag{C.5}$$

Equation C.5 is called the **cyclic permutation property** of the trace operator. Using Equation C.5 we can derive the **trace trick**, which allows us to reorder the scalar inner product $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$:

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \mathrm{tr}(\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}) = \mathrm{tr}(\boldsymbol{x}\boldsymbol{x}^T \boldsymbol{A}) = \mathrm{tr}(\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T) \tag{C.6}$$

References

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (2nd ed.). New York: Springer.

Bertsekas, D. P. and J. N. Tsitsiklis (2008). *Introduction to Probability* (2nd ed.). Belmont, MA: Athena Scientific.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks, Tricks of the Trade, Reloaded*, Volume 7700 of *Lecture Notes in Computer Science*, pp. 430–445. Springer.

Boyd, S. P. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge: Cambridge University Press.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science 16*(3), 199–231.

Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. *Compte rendu des séances de l'acdeémie des sciences 25*, 536–538.

Cover, T. M. and J. A. Thomas (2006). *Elements of Information Theory* (2nd ed.). Hoboken, NJ: Wiley and Sons.

de Finetti, B. (1974). *Theory of Probability: A critical introductory treatment*, Volume 1 and 2. Wiley.

Ellis, S. P. (1998). Instability of least squares, least absolute deviation and least median of squares instability of least squares, least absolute deviation and least median of squares linear regression. *Statistical Science 13*(4), 337–350.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (2003). *Reasoning About Knowledge*. Cambridge, MA: MIT Press.

Galton, F. (1889). *Natural Inheritance*. New York: Macmillan.

Gentle, J. E. (2007). *Matrix Algebra: Theory, Computations, and Applications in Statistics*. New York: Springer.

Ghallab, M., D. Nau, and P. Traverso (2016). *Automated Planning and Acting*. New York: Cambridge University Press.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Haenni, R., J.-W. Romeijn, G. Wheeler, and J. Williamson (2011). *Probabilistic Logics and Probabilistic Networks*. Synthese Library. Dordrecht: Springer.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning* (2nd ed.). Springer Series in Statistics. Springer.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics 35*(1), 73–101.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge: Cambridge University Press.

Kleinbaum, D., L. Kupper, A. Nizam, and E. S. Rosenberg (2014). *Applied Regression Analysis and Other Multivariable Methods* (5th ed.). Cengage Learning.

Kolmogorov, A. N. (1950). *Foundations of the Theory of Probability*. Chelsea Publishing Company.

Manning, C. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

McElreath, R. (2016). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*, Volume 122 of *CRC Texts in Statistical Science*. Boca Raton, FL: Chapman and Hall.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Murphy, R. R. (2019). *Introduction to AI Robotics* (2nd ed.). Cambridge, MA: MIT Press.

Pollard, D. (2002). *A User's Guide to Measure Theoretic Probability*. Cambridge: Cambridge University

Press.

Prince, S. J. D. (2012). *Computer Vision: Models, Learning, and Inference.* New York: Cambridge University Press.

Russell, S. and P. Norvig (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River: Prentice Hall.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal 3*, 210–229.

Savage, L. J. (1954). *Foundations of Statistics.* New York: Wiley.

Seidenfeld, T., M. J. Schervish, and J. B. Kadane (2001). Improper regular conditional distributions. *The Annals of Probability 29*(4), 1612–1624.

Spirtes, P., C. Glymour, and R. Scheines (2000). *Causation, Prediction, and Search.* Cambridge, MA: MIT Press.

Stigler, S. M. (1986). *The History of Statistics: The Measurement of Uncertainty Before 1900/.* Cambridge, MA: Harvard University Press.

Sullivan, A. M. (1970). Measurement. In *Selected Lyrics and Sonnets.* New York: Thomas Y. Crowell.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA: MIT Press.

Thomson, W. (1889). Electrical units of measurement. In *Popular Lecturers and Addresses.* New York: Macmillan. A lecture originally delivered on May 3, 1883, at the Institition of Civil Engineers.

Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference.* New York: Springer.

Wheeler, G. (2018). Bounded rationality. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2018 ed.). Metaphysics Research Lab, Stanford University.

Alphabetical Index