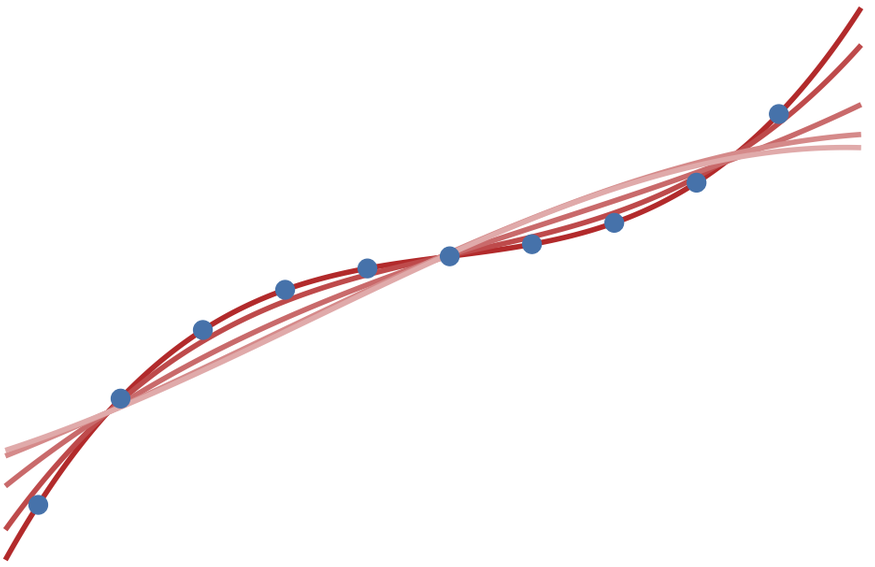
Theoretical Foundations *of* Data Science

Notes for DSC40A

Sawyer Jack Robertson



Preface

This book contains notes for DSC40A taught at UC San Diego during Summer Session 2025. Please keep in mind that this will(!) contain typos and errors throughout. If you find any you are welcome to let me know at s5robert (at) ucsd (dot) edu.

`The version you are reading was compiled on October 7, 2025.`

Table of Contents

Chapter 1. Foundational Concepts 5

1.1. The modeling method 5

1.2. The constant model and loss functions 11

1.3. Vector-valued features and targets 17

1.4. Exercises 23

Chapter 2. The Linear Model 33

2.1. Simple Linear Regression: Scalar Features, Scalar Targets 33

2.2. Warmup: The constant model revisited 40

2.3. Multiple Linear Regression: Vector Features, Scalar Targets 43

2.4. The General Linear Model: Vector Features, Vector Targets 52

2.5. Exercises 56

Chapter 3. More on Modeling 68

3.1. Polynomial Regression and Interactions 68

3.2. Convexity & Gradient Descent 75

3.3. Regularization: Ridge and Lasso Regression 84

3.4. Bonus: Constrained optimization 91

3.5. Exercises 95

Chapter 4. Modeling with Probability 104

4.1. Sample Spaces and Probability Measures 104

4.2. Counting and Combinatorics 107

4.3. Independence and Conditional Probability 112

4.4. Naïve Bayes Classifiers 118

4.5. Exercises 123

Chapter 5. Appendices 132

A. Calculus Background 132

B. Linear Algebra Background 133

Chapter 6. Index 135

1 Foundational Concepts

> Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question...
>
> - *John W. Tukey*

This chapter will cover several foundational concepts and themes which will permeate the rest of our journey. We will begin by covering the "modeling method," which is intended to serve as a guiding beacon for the way data scientists approach their work. Much in the same way that a biologist might utilize the scientific method to understand the root cause of some particular disease, a data scientist utilizes the method contained herein to build a neural network to determine whether a plant leaf is diseased or healthy.

Then, we will spend some time focusing on the constant model, which is probably the simplest model and this will give us an opportunity to explore different choices of loss functions as well.

1.1 The modeling method

Every scenario we encounter as data scientists consists of a few fundamental ingredients. Consider the following example: Annabeth is a biologist working at a company that studies infectious diseases in mice. She wants to understand how the mice respond to different drug dosages. She measures the dosages in units of *milligram per gram of body weight*, and she measures the mouse's response in terms of *hours survived after infection*. In this example, Annabeth would like to build a model that, given a specific dosage, predicts the number of hours a mouse will survive after infection.

The dosages are what we call **input variables**, also known as **features**, and are the data or attributes which are used to make predictions. They represent the independent variables in the context of a mathematical model or statistical analysis. On the other hand, **output variables**, also known as *targets*, are the data or attributes that the model aims to predict. They represent the dependent variables in the context of a mathematical model. In this example, mouse survival time is considered the output variable.

Input and output variables come in all shapes and sizes. **Numerical variables** consist of numbers (or vectors of numbers), and include such things as the dosages we saw earlier; other examples include things like the height of a person, the weight of an acorn, or the brightness of a pixel on a screen. **Categorical variables** correspond to discrete units which originate from a fixed list; these include examples like the color or breed of a cat, the genre of a movie, or the major of a college student. **Binary variables** are a subclass of categorical variables, and correspond to situations where the variable can take on one of two possibilities, e.g., true/false, diseased/healthy, success/failure. By identifying the input and output variables, we have completed **step one** of the modeling method.

? Describe a scenario where the input variables consist of multiple different types.

A **model** is a mathematical function that maps features to targets based on a set of parameters, aiming to capture the underlying relationship between them. **Parameters**,

also known as *weights*, are the numerical values which are used to define the model itself; these are distinct from the input variables, but may depend on them. One can think of the weights of a model as the dials and knobs that are adjusted in response to the input data in order to get an accurate response. **Model training** is, quite simply, the process of tuning the weights of a model in order to improve the accuracy of the model overall.

Returning to our example, Annabeth decides to use a model to understand the relationship between mouse survival and drug dosages. She decides to use the following approach: letting x denote the drug dosage in units of milligram per gram of body weight, she wishes to model the survival time of a mouse by $f(c;x)$, where

$$f(c;x) = cx,$$

for some $c \in \mathbb{R}$. This is an example of a **linear model**, about which we will learn more in the next chapter. The model consists of a single parameter: c. By selecting a model, we have completed **step two** of the modeling method.

? What are some examples of different models Annabeth could use?

But we have a dilemma: How should Annabeth select c in order to produce an accurate prediction of mouse survival time? She will need to use a loss function. A **loss function** is a function which is used in machine learning to quantify the difference between a model's predicted output and the actual target values. By assigning a numerical value to prediction errors (sometimes called "costs"), it guides the selection of weights to improve the model's performance. Examples include: mean-squared error (MSE), absolute loss, cross-entropy loss, and more.

To keep things as simple as possible for now, Annabeth chooses to use the **square loss**, which is set up as follows. Let c denote a particular choice of the model parameter, let x denote the drug dosage, and let y denote its *actual* survival time. We write

$$L(c;(x,y)) = (y - cx)^2.$$

In this notation, we write $L(c;(x,y))$ to mean "the loss associated with a choice of the parameter when the observation (x,y) is given." The semicolon serves to separate the parameter from the input-output pair being used when evaluating the error of the prediction, which in this case is cx. Notice that when $y \approx cx$, i.e., the predicted survival time is close to c(dosage), then the loss will be small; otherwise, it will grow quadratically in terms of the difference. The choice of a loss function depends on the nature of the problem (e.g., regression vs. classification) and the desired behavior of the model, such as robustness to outliers or interpretability of the optimization process. By choosing a loss function, we have completed **step three** of the modeling method.

If Annabeth has a single mouse in her study, with observed input-output variables (x_1, y_1), then finding c which minimizes $L(c;(x,y))$ can be done with some calculus, as follows. We begin by differentiating $L(c;(x_1,y_1))$ with respect to c:

? Why do we use d instead of ∂ here?

$$\begin{aligned}\frac{\mathrm{d}L}{\mathrm{d}c} &= \frac{\mathrm{d}}{\mathrm{d}c}\big(y_1 - cx_1\big)^2 \\ &= 2\big(y_1 - cx_1\big) \cdot \frac{\mathrm{d}}{\mathrm{d}c}\big(y_1 - cx_1\big) \\ &= 2\big(y_1 - cx_1\big) \cdot (-x_1).\end{aligned}$$

Simplifying, we have:

$$\frac{\mathrm{d}L}{\mathrm{d}c} = -2x_1\big(y_1 - cx_1\big).$$

To find the value of c that minimizes the loss, we search for a critical point where $\frac{\mathrm{d}L}{\mathrm{d}c} = 0$. Once we do this, we are solving an equation for a *particular* value of c - let's denote it c^*, and solve

$$-2x_1(y_1 - c^*x_1) = 0.$$

Since $x_1 \neq 0$ (we assume the dosage is nonzero), we can divide through by $-2x_1$, and yield:

$$y_1 - c^*x_1 = 0.$$

Rearranging gives:

$$c^* = \frac{y_1}{x_1}.$$

Try to replicate this calculation for a different choice of L!

Next, we note that

$$\frac{\mathrm{d}^2L}{\mathrm{d}c^2} = \frac{\mathrm{d}}{\mathrm{d}c}(-2x_1(y_1 - cx_1)) = 2x_1^2 > 0$$

Since the second derivative of L is always positive, by the second derivative test (see Appendix A), we may conclude that the value c^* which minimizes the loss for this single mouse is the ratio of the actual survival time y_1 to the dosage of the mouse x_1.

But what if Annabeth's experiment consists of, say, ten mice instead of a single one? Our choice of c needs to be informed by all of the data at hand in order to be as accurate as possible. The dataset of features used to inform the choice of parameters in a model is sometimes called the **training data**, and is distinguished from **testing data**, which is the dataset of examples that a data scientist will usually set aside during the training process in order to evaluate the performance of the model. In this example, the training data might consist of ten dosages $x_1, x_2, \ldots, x_{10}$, each with their corresponding survival times $y_1, y_2, \ldots, y_{10}$.

In order to use *all* of the training data, we use our loss function to build what is called a risk function. A **risk function** is any function used to estimate the loss incurred by a model over an entire dataset. In the same manner that there are many choices for different loss functions, there are also many choices for risk functions. In this section we focus exclusively on the **empirical risk function**, which is just the average over all of the losses in the training set. In Annabeth's setting, we write

$$\begin{aligned} R(c; \{(x_i, y_i)\}_{i=1}^{10}) &= \frac{1}{10}\sum_{i=1}^{10} L(c; (x_i, y_i)) \\ &= \frac{1}{10}\left((y_1 - cx_1)^2 + (y_2 - cx_2)^2 + \ldots + (y_{10} - cx_{10})^2\right). \end{aligned}$$

We use the notation $R(c; \{(x_i, y_i)\}_{i=1}^{10})$ to mean "the risk associated with the parameter choice c and the input and output pairs given by

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_{10}, y_{10})\}."$$

As long as we understand that $\{(x_i, y_i)\}_{i=1}^{10}$ are the training data, we may write $R(c)$ for short. In this context, finding the "best" parameter c for modeling the entire dataset can be recast as finding c which minimizes $R(c)$. We can approach this via a similar

derivation as before, as follows. We first compute the derivative of $R(c)$ with respect to c:

$$\begin{aligned}\frac{\mathrm{d}R}{\mathrm{d}c} &= \frac{1}{10}\sum_{i=1}^{10}\frac{\mathrm{d}}{\mathrm{d}c}(y_i - cx_i)^2 \\ &= \frac{1}{10}\sum_{i=1}^{10}2(y_i - cx_i)\cdot\frac{\mathrm{d}}{\mathrm{d}c}(y_i - cx_i) \\ &= \frac{1}{10}\sum_{i=1}^{10}2(y_i - cx_i)\cdot(-x_i).\end{aligned}$$

Simplifying, we have:

$$\begin{aligned}\frac{\mathrm{d}R}{\mathrm{d}c} &= -\frac{2}{10}\sum_{i=1}^{10}x_i(y_i - cx_i) \\ &= -\frac{2}{10}\sum_{i=1}^{10}(x_iy_i - cx_i^2).\end{aligned}$$

Factoring out the summation:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = -\frac{2}{10}\left(\sum_{i=1}^{10}x_iy_i - c\sum_{i=1}^{10}x_i^2\right).$$

We set $\frac{\mathrm{d}R}{\mathrm{d}c} = 0$ to find the critical point. Once again, this is now an equation for c, so we use the notation c^* to emphasize that we are searching for a special value. Although this expression might look unfriendly, keep in mind– we are solving for c^*, which appears as a single term in the middle:

$$\begin{aligned}0 &= -\frac{2}{10}\left(\sum_{i=1}^{10}x_iy_i - c^*\sum_{i=1}^{10}x_i^2\right). \\ \iff 0 &= \sum_{i=1}^{10}x_iy_i - c^*\sum_{i=1}^{10}x_i^2. \\ \iff c^* &= \frac{\sum_{i=1}^{10}x_iy_i}{\sum_{i=1}^{10}x_i^2}. \qquad (1)\end{aligned}$$

What changes if there are n mice instead of 10?

We leave it as an exercise to verify that $\frac{\mathrm{d}^2R}{\mathrm{d}c^2} > 0$ for our dataset. Thus, the value c^* that minimizes the empirical risk is given by the ratio of the sum of the products of x_i and y_i to the sum of the squares of x_i. We can bring this example to life in Fig. 1 by using Annabeth's fictional dataset. Having found c^*, we have completed **step four** of the modeling method, and are finally ready to state it in full, which we include below.

The Modeling Method

1. Identify your input and output variables.
2. Choose a model.
3. Choose a loss function and a risk function.
4. Find a minimizer of the risk function.

x_i	y_i
4.37	10.46
9.56	24.43
7.59	18.51
6.39	15.50
2.40	6.25
2.40	4.10
1.52	2.08
8.80	21.43
6.41	15.01
7.37	18.75

$c^* \approx 2.45$

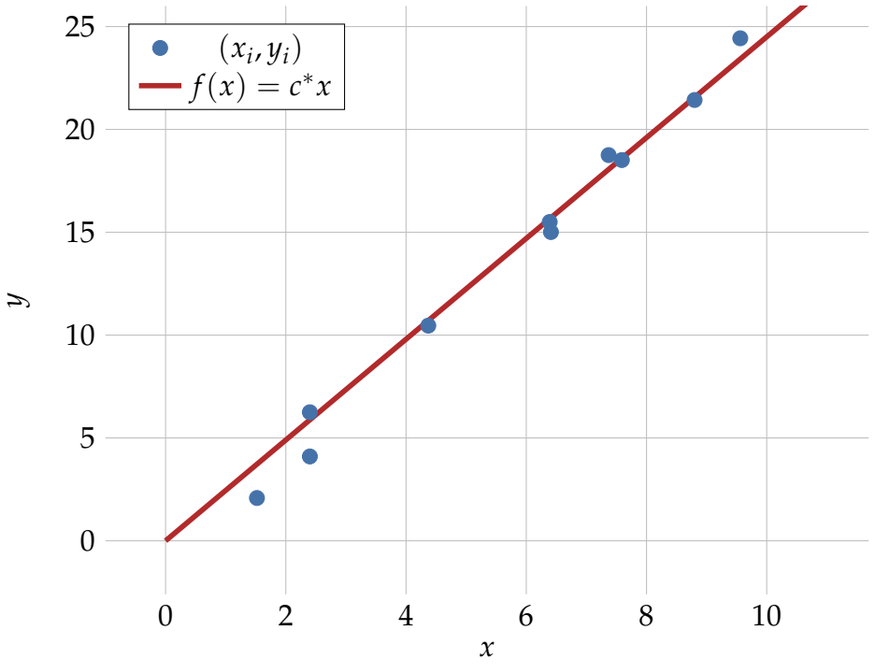


Figure 1: **(left)** Annabeth's dataset for ten mice. The dosages x_i are measured in units of mg per g of body weight, and the survival times y_i are measured in hours post infection. The choice of c which minimizes the empirical risk is approximately 2.45. **(right)** A scatterplot of the training data $\{(x_i, y_i)\}_{i=1}^{10}$ situated in the same axes as the function $f(x) = c^* x$, which corresponds to the predicted survival times for the dosages.

Example 1.1.1 *The riper the fruit*

Marcus is a farmer who grows mangoes. He wants to predict how ripe a fruit will be based on the number of days since it was harvested. He measures ripeness on a numeric scale between 0 (unripe) and 10 (perfectly ripe). Marcus believes that the ripeness y of a fruit depends logarithmically on the number of days x since harvest. He chooses a model where ripeness is given by:

$$f(c; x) = c \ln x,$$

where $c \in \mathbb{R}$ is a parameter that Marcus will determine based on his data.
Marcus collects data for ten fruits. The number of days since harvest (x_i) and the observed ripeness (y_i) are shown in the table below.

x_i (days)	y_i (ripeness)
1.2	1.5
2.5	2.8
3.1	3.2
4.8	4.0
5.6	4.5
6.0	4.8
7.2	5.5
8.1	6.0
9.0	6.3
10.5	6.9

To determine the optimal value of c, Marcus chooses the square loss:

$$L(c;(x,y)) = \big(y - c\ln x\big)^2.$$

Using the square loss, the empirical risk function for Marcus's data is:

$$R(c) = \frac{1}{10}\sum_{i=1}^{10}\big(y_i - c\ln x_i\big)^2.$$

We compute the derivative of $R(c)$ with respect to c, as follows:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = -\frac{2}{10}\sum_{i=1}^{10}\ln x_i\big(y_i - c\ln x_i\big).$$

Setting $\frac{\mathrm{d}R}{\mathrm{d}c} = 0$, we solve for c^*:

$$\begin{aligned}\sum_{i=1}^{10}\ln x_i\cdot y_i &= c^*\sum_{i=1}^{10}(\ln x_i)^2\\ \iff c^* &= \frac{\sum_{i=1}^{10}\ln x_i\cdot y_i}{\sum_{i=1}^{10}(\ln x_i)^2}.\end{aligned}$$

Using Marcus's data:

$$c^* = \frac{(1.5\cdot\ln 1.2) + (2.8\cdot\ln 2.5) + \cdots + (6.9\cdot\ln 10.5)}{(\ln 1.2)^2 + (\ln 2.5)^2 + \cdots + (\ln 10.5)^2} \approx 2.81.$$

The following `Python` code generates a scatterplot of Marcus's dataset and plots the best-fit logarithmic model found above.

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Data: days since harvest (x) and observed ripeness (y)
x = np.array([1.2, 2.5, 3.1, 4.8, 5.6, 6.0, 7.2, 8.1, 9.0, 10.5])
y = np.array([1.5, 2.8, 3.2, 4.0, 4.5, 4.8, 5.5, 6.0, 6.3, 6.9])

# Compute c*
log_x = np.log(x)
c_star = np.sum(log_x * y) / np.sum(log_x**2)

# Define the model
def model(x):
    return c_star * np.log(x)

# Plot data and the model
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', label='Observed Data')
plt.plot(np.linspace(1, 11, 100), model(np.linspace(1, 11, 100)),
```

```
                color='red', label=f'Model: $f(x) = {c_star:.2f} \\ln x$')
plt.xlabel('Days since harvest ($x$)')
plt.ylabel('Ripeness ($y$)')
plt.title('Ripeness Prediction')
plt.legend()
plt.grid(True)
plt.show()
```

1.2 The constant model and loss functions

Key Idea

The constant model is given by $f(c; x) = c$ for some $c \in \mathbb{R}$. Note that c may depend on the training data, but f always outputs the same value regardless of input.

In this section we focus on the simplest model of all: the **constant model**. Don't be fooled! The constant model plays a foundational role in data science. Training the constant model amounts to answering the question: "What is the best *single* value to represent or summarize all of my data?" This simplicity makes the constant model an essential starting point for understanding more complex models, as it allows us to more easily understand questions such as:

- How do different loss functions L lead to changes in the minimizer c^* for the empirical risk?
- How is the constant model's minimizer c^* impacted by outliers or other unusual patterns in the training data?
- How do transformations in our training data, like scaling and translations, change c^*?

In this section we will try our best to find some answers to these questions. We begin with an exploration of different loss functions. In Section 1.1, we used the **square loss**, which more descriptively may be written

$$L_{\text{sq}}(c; (x, y)) = (y - f(c; x))^2$$

where x is an input variable, y is an output variable and $f(c; x)$ is our model. We chose this loss function because it is easy to do calculus with (this will be a theme that continues throughout the book), and in turn, it's relatively easy to find its minimizer. In Annabeth's example, the training data consisted of *both* some input variables x_i as well as their actual output variables y_i. This is an example of **supervised learning**, where labels or targets are known while we train the model (sometimes described as "at training time").

In the setting of the constant model, we will assume our training data consist of the values $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}$ for each $1 \leq i \leq n$. Our training data does not come pre-equipped with targets; rather, our goal is to interpolate the training data with a single

constant. This is called **unsupervised learning**, where no examples of targets or output variables are provided to the data scientist at the time they train their model. Thus, in our setting, the square loss takes the form

$$L_{\text{sq}}(c; x_i) = (c - x_i)^2.$$

Another type of loss function which is of interest to us will be the **absolute loss**, which is given by

$$L_{\text{abs}}(c; x_i) = |c - x_i|.$$

We may also consider the p-**loss**, where $1 \leq p < \infty$ is a fixed parameter (and is not trained or subject to learning):

$$L_{\text{p}}(c; x_i) = |c - x_i|^p.$$

Note that when $p = 1$, $L_{\text{p}} = L_{\text{abs}}$ and when $p = 2$, $L_{\text{p}} = L_{\text{sq}}$. We begin by exploring the first question, and obtain a formula for the minimizer c^* of the empirical risk in the case of the squared loss.

Note: the terms which appear to the right of the semi-colon are whatever training data is available; thus, you may see (x_i, y_i) or simply x_i depending on whether the model is "supervised."

? The square loss penalizes large errors more heavily than small ones due to squaring. Why might this be useful for certain applications, like financial forecasting?

Theorem 1.2.1

The minimizer c^* for the empirical risk associated with the square loss is given by the mean of the training data, i.e.,

$$c^* = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2}$$

We will use the notation $\overline{x}$ to denote the mean of the scalars $x_1, \ldots, x_n$. This notation will also be applied to vectors, matrices, and beyond, but will always have a shared meaning: take the list of objects we're considering, add them up, and divide by the total number. The proof of Theorem 1.2.1 should feel fairly natural after the examples in the preceding section.

Proof

The empirical risk for the square loss is given by:

$$R(c) = \frac{1}{n} \sum_{i=1}^{n} (c - x_i)^2.$$

As usual, we compute its derivative with respect to c,

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{1}{n} \sum_{i=1}^{n} 2(c - x_i).$$

Simplifying:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{2}{n} \sum_{i=1}^{n} (c - x_i).$$

We set $\frac{\mathrm{d}R}{\mathrm{d}c} = 0$ to find the critical point.

$$\sum_{i=1}^{n}(c^* - x_i) = 0.$$

Expanding and rearranging, we have:

$$nc^* = \sum_{i=1}^{n} x_i \iff c^* = \overline{x}.$$

Thus, there is a critical point c^* corresponding to the mean of the training data. We can check that it is a true minimizer by using the second derivative test. The second derivative of $R(c)$ with respect to c is:

$$\frac{\mathrm{d}^2R}{\mathrm{d}c^2} = \frac{2}{n}\sum_{i=1}^{n} 1 = \frac{2}{n} \cdot n = 2.$$

Since $\frac{\mathrm{d}^2R}{\mathrm{d}c^2} > 0$, the critical point $c^* = \overline{x}$ is indeed a global minimizer.

Next we consider the question: what changes if instead we consider the absolute loss in place of the squared loss? To do this, we begin with an example which explores the associated empirical risk.

🍊 **Example 1.2.2** ***Sarah's coffee shop***

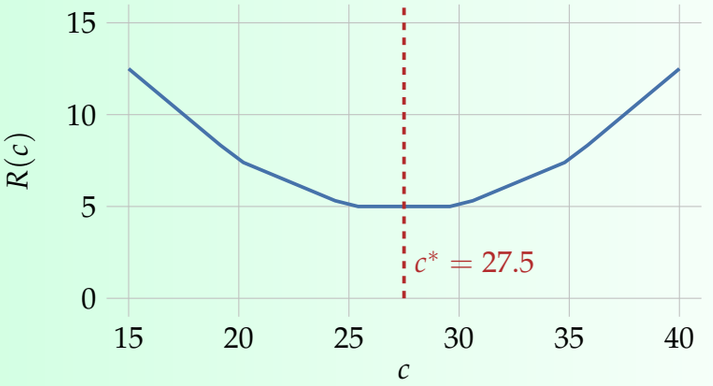
Sarah is a data analyst tracking the daily number of customers visiting a small café over four days. To simplify her analysis, she decides to use the constant model to summarize the data. Her dataset for the number of daily customers is given below:

x_1	x_2	x_3	x_4
20	30	25	35

To understand how it varies as c changes, Sarah wants to plot the empirical risk $R(c; \{x_i\}_{i=1}^4)$ associated with the absolute loss:

$$R(c) = \frac{1}{4}\sum_{i=1}^{4} |c - x_i| = \frac{1}{4}\left(|c-20| + |c-30| + |c-25| + |c-35|\right).$$

She plots $R(c)$ as a function of c, and highlights a point c^* which minimizes the empirical risk.



From the plot, she sees that the empirical risk $R(c)$ has many different minimizers in the range $25 \leq c \leq 30$; she may take, for example $c^* = 27.5$.

The preceding example suggests that finding the minimizer of the empirical risk associated to the absolute loss will require more care, and can in fact be nonunique, unlike the mean of the data. Our next fact shows that the minimizer we are looking for is actually the median of our training data.

Theorem 1.2.3

The minimizer c^* for the empirical risk associated with the absolute loss is given by *any* median of the training data, i.e.,

$$c^* = \text{median}(\{x_i\}_{i=1}^n).$$

Notably, c^* is not always unique.

Proof

We can assume our data x_i is numbered so that the data is arranged from least to greatest; that is, we will assume

$$x_1 \leq x_2 \leq \ldots \leq x_{n-1} \leq x_n.$$

Note that the absolute loss is a piecewise function, i.e.,

$$L_{\text{abs}}(c; x_i) = \begin{cases} (c - x_i) & \text{if } c \geq x_i \\ -(c - x_i) & \text{if } c < x_i \end{cases}. \tag{3}$$

It is important to point out that $L(c; x_i)$ is continuous but *not differentiable* when $c = x_i$, where it has a "corner;" so, when applying techniques from calculus to find the minimizer of the empirical risk, we need to exercise caution. Therefore, the derivative of $|c - x_i|$ is given by:

$$\frac{\mathrm{d}}{\mathrm{d}c} L_{\text{abs}}(c; x_i) = \begin{cases} 1 & \text{if } c > x_i, \\ \text{DNE} & \text{if } c = x_i, \\ -1 & \text{if } c < x_i. \end{cases}$$

The empirical risk $R(c; \{x_i\}_{i=1}^n)$ is given by the expression

$$R(c) = \frac{1}{n} \sum_{i=1}^{n} |c - x_i| = \frac{1}{n} \left(|c - x_1| + |c - x_2| + \ldots + |c - x_n| \right).$$

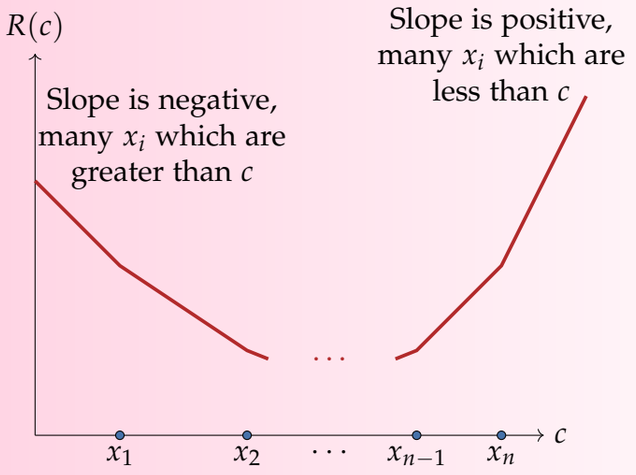
This consists of at most $n+1$ piecewise-linear segments, which are linked together. It is differentiable everywhere except at the points $c = x_1, x_2, \ldots, x_n$ where it has corners. By the critical point theorem (see Theorem A.1), a minimizer of $R(c)$ must occur where $R'(c) = 0$ or where $R'(c)$ does not exist. Note that the second derivative of $R(c)$, when it exists, is zero. Therefore after finding the critical points we must inspect them to identify those which correspond to minimizers of $R(c)$. To find these points, assuming c does not equal one of the values x_i, we can find the derivative of $R(c)$ using Eq. (3) as follows:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{1}{n} \left(\sum_{x_i \leq c} (1) + \sum_{x_i > c} (-1) \right).$$

In other words, $\frac{\mathrm{d}R}{\mathrm{d}c}$ decreases by $1/n$ for each data point x_i which is less than c, and increases by $1/n$ for each data point x_i which is greater than c. That is,

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{1}{n} \left(\#\{x_i : x_i \leq c\} - \#\{x_i : x_i > c\} \right).$$

If there exist some values of c where there are *exactly* as many data points which are larger than c as those that are less than c, then $\frac{\mathrm{d}R}{\mathrm{d}c} = 0$. Sometimes these values occur at a "corner," where $\frac{\mathrm{d}R}{\mathrm{d}c}$ does not exist, but these will still correspond to the minimizers of $R(c)$. See the illustration below to get a feeling for what this argument is telling us.



The minimizers correspond to the medians of the dataset, which do not have to be unique. For example, the empirical risk of the dataset $\{1, 2, 3, 4\}$ has minimizers for any $2 \leq c^* \leq 3$. On the other hand, the empirical risk of the dataset $\{1, 2, 3\}$ has exactly one minimizer when $c^* = 2$.

When $p \neq 1, 2$ the minimizer(s) for the empirical risk L_p need not have "nice" closed-form solutions, although this is occasionally the case for some values of p. Thus, the absolute loss and squared loss correspond to certain instances where the minimizers of the empirical risk are more readily obtained and can be discussed at length.

Note that both of the preceding discussions consider the case where the features x_i are numerical variables taking the form of arbitrary real numbers. As a third example, we consider a more specialized setting. We assume our data $x_i \in \{0, 1\}$, and represent binary categorical variables (such as successes and failures, or boolean data). The **entropy loss** is defined for parameters $c \in (0, 1)$ as follows:

$$L_H(c; x_i) = -\big[x_i \ln c + (1 - x_i) \ln(1 - c)\big].$$

The letter H is often used to denote entropy, a concept which is borrowed from the fields of information theory and physics. Note that c may be interpreted as a probability that the input data x_i achieves $x_i = 1$.

Theorem 1.2.4

Assuming $x_i \in \{0, 1\}$, the minimizer c^* for the empirical risk associated with the entropy loss is given by the mean of the training data, i.e.,

$$c^* = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{4}$$

We leave the proof of this fact as an exercise. Note that if all of our training data are the same, the empirical risk has a "minimum value" of $-\infty$ because of the asymptote of $\ln x$ as $x \to 0^+$.

We now turn to the second question which was highlighted at the beginning of the section:

How is the constant model's minimizer c^* impacted by outliers or other unusual patterns in the training data?

how are the minimizers c^* affected by outlier data points? To get a sense of this, suppose we come across the following hypothetical dataset:

$$\{\underbrace{0, 0, \ldots, 0}_{\text{1,000 times}}, 1000\}.$$

This dataset consists of the value 0 which has been observed $1,000$ times, followed by a single outlier: $x_{1,001} = 1,000$. If we wished to use the constant model to interpret out data, the minimizer c^*_{sq} of empirical risk for the squared loss would give us

$$c^*_{\text{sq}} = \overline{x} = \frac{1}{1001}(0 + 0 + \ldots + 0 + 1000) \approx 1.$$

So while the mean of the first $1,000$ data points is zero, the addition of the outlier $x_{1,001} = 1,000$ causes the mean to jump one unit. On the other hand, the minimizer c^*_{abs} for the empirical risk associated with the absolute loss would be given by

$$c^*_{\text{abs}} = \text{median}\{0, 0, \ldots, 0, 1000\} = 0.$$

This demonstrates that c^*_{abs} and c^*_{sq} respond differently to outliers, behavior which we summarize below.

Key Idea

The minimizer c^*_{sq} for the empirical risk associated with the squared loss is more *sensitive* to outliers than the minimizer c^*_{abs} for the empirical risk associated with the absolute loss.

? If you are trying to model housing prices in a neighborhood with the constant model, when would it be preferable to use the squared loss? The absolute loss?

Sensitivity to outliers is neither entirely good nor bad; it often depends heavily on the context of one's model. For example, in a scenario where outliers are rare but critically important (detecting irregular heart patterns or seismic activity), developing a model which easily responds to these is important. On the other hand, in scenarios where outliers arise frequently due to noisy data (analyzing long-term weather patterns, or tracking animals via radio telemetry), developing models which handle outliers in stride without breaking down can be useful.

Finally, we can turn to the third question which was posed at the beginning of the section:

How do transformations in our training data, like scaling and translations, change c^*?

There are *many* different ways we could approach a question like this, so to conclude this section, we investigate an example below and then explore extensions of this theme in the exercises to follow.

Example 1.2.5 ***Scaling and translating***

Let $\{x_i\}_{i=1}^n$ be a fixed dataset of real numbers, and suppose we fix two parameters $a, b \in \mathbb{R}$ which are separate from our training data. Then, we define

$$y_i = ax_i + b, \quad 1 \leq i \leq n.$$

How does the minimizer c^*_{sq} for the empirical risk associated with the squared loss change under this transformation? We know from Theorem 1.2.1 that the minimizer will be $\overline{y}$, and thus with some algebraic manipulations, we have

$$\begin{aligned} \overline{y} &= \frac{1}{n}\sum_{i=1}^{n}(ax_i + b) \\ &= a\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + b\left(\frac{1}{n}\sum_{i=1}^{n}(1)\right) \\ &= a\overline{x} + b. \end{aligned}$$

1.3 Vector-valued features and targets

We wish to emphasize early on that our jobs as data scientists will require familiarity with vector-valued features and targets in addition to the scalar-valued features and targets encountered already in the preceding sections. With this in mind, we will revisit two of the one-parameter models we have already seen: the simplified linear model from Section 1.1 and the constant model in Section 1.2. These methods require familiarity with the notion of vector lengths in higher dimensions. In this book, we will denote vectors

using the notation $\vec{x} \in \mathbb{R}^d$. We will denote the coordinates of vectors in the form $\vec{x}^{(i)}$ where $1 \leq i \leq d$. Matrices will generally be denoted with bold letters, for example, $\mathbf{X} \in \mathbb{R}^{d \times d}$, and their entries will be denoted $\mathbf{X}^{(i,j)}$.

To start off, we revisit the idea of training data. In the preceding sections, we considered $x_1, \ldots, x_n \in \mathbb{R}$. It is straightforward enough to generalize this by considering instead vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$ as before. As a matter of convention, we use the letter $d \geq 1$ to denote the **feature dimension**, which refers to the dimension of the vector space from which are samples are being drawn. Some examples of vector-valued features include:

- An array of 1000 black-and-white 28×28 images, where $d = 784$ and $n = 1000$;
- A collection of average daily temperatures for each week in a year, where $d = 7$ and $n = 52$;
- A dataset of average daily stock prices of 500 companies for each day of 2020, where $d = 366$ and $n = 500$.

The **target dimension** refers to the dimension of the vector space from which targets or labels are obtained. In this book we will use $p \geq 1$ to denote the target dimension. Some examples of vector targets include, using the previous examples,

- The probabilities of each image belonging to each of the classes $\{\text{cat}, \text{dog}, \text{horse}\}$, where $p = 3$;
- The average temperature on each day of the week across an entire year, where $p = 7$;
- The average return on $\$1,000$ invested on January 1, 2020 in each quarter of the calendar year 2020, where $p = 4$.

? In the two examples below, revisit the modeling method at the end of Section 1.1 and describe each step in detail.

Example 1.3.1 *The vector-valued constant model*

Let $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ denote a dataset of training examples in $\mathbb{R}^d$. Suppose we wish to use the model

$$f(c; \vec{x}) = c \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = c\vec{\mathbf{1}}_d,$$

where $\vec{\mathbf{1}}_d$ is the vector of all ones in $\mathbb{R}^d$. We refer to this as the **vector-valued constant model** where the feature and target dimensions are the same. In this setting, the square loss takes on a slightly different notion, where instead we measure the length squared:

$$L_{\text{sq}}(c; \vec{x}_i) = \|c\vec{\mathbf{1}}_d - \vec{x}_i\|^2 = \sum_{s=1}^{d} (c - \vec{x}_i^{(s)})^2.$$

The empirical risk can then be written

$$R(c) = \frac{1}{n}\sum_{i=1}^{n} \|c\vec{\mathbf{1}}_d - \vec{x}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{d}(c - \vec{x}_i^{(s)})^2.$$

As usual, to find the minimizer c^*, we take the derivative of $R(c)$ with respect to c:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{d} 2(c - \vec{x}_i^{(s)}) = \frac{2}{n}\sum_{i=1}^{n}\sum_{s=1}^{d}(c - \vec{x}_i^{(s)}).$$

Setting $\frac{\mathrm{d}R}{\mathrm{d}c} = 0$ yields

$$\sum_{i=1}^{n}\sum_{s=1}^{d}(c^* - \vec{x}_i^{(s)}) = 0 \implies ndc^* = \sum_{i=1}^{n}\sum_{s=1}^{d}\vec{x}_i^{(s)},$$

which implies

$$c^* = \frac{1}{n\,d}\sum_{i=1}^{n}\sum_{s=1}^{d}\vec{x}_i^{(s)}.$$

Finally, we confirm that this critical point is a minimizer by checking the second derivative. Note that

$$\frac{\mathrm{d}^2R}{\mathrm{d}c^2} = \frac{\mathrm{d}}{\mathrm{d}c}\Big[\tfrac{2}{n}\sum_{i,s}(c - \vec{x}_i^{(s)})\Big] \;=\; \frac{2}{n}\sum_{i=1}^{n}\sum_{s=1}^{d}1 \;=\; \tfrac{2}{n}\,(n\,d) \;=\; 2d,$$

which is strictly positive for $d \geq 1$. Therefore, by the second derivative test, c^* is a global minimizer. Thus, the minimizer of the empirical risk for the squared loss in the constant vector model is simply the average over all of the coordinates in the dataset.

🍊 Example 1.3.2 *Drone delivery*

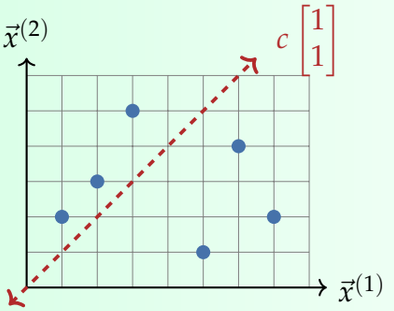
Imagine a drone delivery company that serves customers distributed across a two-dimensional region. We represent each customer's location by coordinates

$$\vec{x}_i = (\vec{x}_i^{(1)}, \vec{x}_i^{(2)})^\top \in \mathbb{R}^2.$$

Our goal is to choose one fixed drone station location along a central street, using the constant model, in such a way that the average squared distance to all customers is minimized. Suppose we have six customers located at the points

$$\{(1,2),\ (2,3),\ (3,5),\ (6,4),\ (7,2),\ (5,1)\}.$$

The following figure provides a visual representation of these locations:



We adopt the constant model

$$f(c;\vec{x}) = c\,\vec{\mathbf{1}}_2 = c\begin{bmatrix}1\\1\end{bmatrix},$$

which, in this context, means selecting a single location (c,c) that best serves all customers. The empirical risk, defined as the average squared distance to the customers, is

$$R(c) = \frac{1}{6}\sum_{i=1}^{6}\|(c,c)-\vec{x}_i\|^2.$$

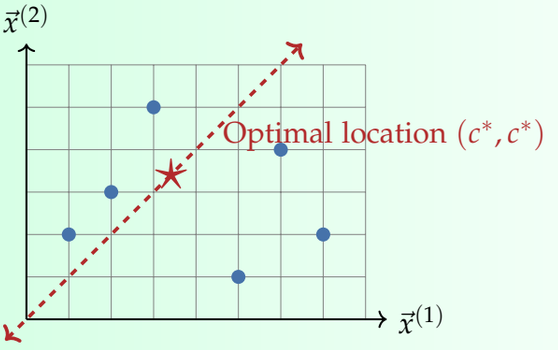
Minimizing $R(c)$ yields the optimal value

$$c^* = \frac{1}{2\times 6}\sum_{i=1}^{6}\left(x_i^{(1)}+x_i^{(2)}\right).$$

A direct computation shows

$$c^* = \frac{(1+2+3+6+7+5)+(2+3+5+4+2+1)}{2\times 6} \approx 3.41.$$

Therefore, the optimal station location is approximately at $(3.41, 3.41)$, as illustrated below:



We may also consider the following extension of the simplified linear model $f(c;x) = cx$ which we saw in Annabeth's experiment with mice and drug development.

Example 1.3.3 *The vector-valued simplified linear model*

Let $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ denote a dataset in $\mathbb{R}^d$, and suppose we pair each sample with a label $\vec{y}_i \in \mathbb{R}^d$. Suppose we wish to use the model

$$f(\vec{x}) = c\vec{x}$$

where $c \in \mathbb{R}$. We refer to this as the **vector-valued simplified linear model** where the feature and target dimensions are the same. Similar to the preceding example, we choose to use the loss function

$$L_{\text{sq}}(c; \vec{x}_i) = \|c\vec{x}_i - \vec{y}_i\|^2 = \sum_{s=1}^{d} (c\vec{x}_i^{(s)} - \vec{y}_i^{(s)})^2.$$

The empirical risk is

$$R(c) = \frac{1}{n}\sum_{i=1}^{n} \|c\vec{x}_i - \vec{y}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{d} (c\,\vec{x}_i^{(s)} - \vec{y}_i^{(s)})^2.$$

To find the minimizer, we compute the derivative:

$$\frac{\mathrm{d}R}{\mathrm{d}c} = \frac{2}{n}\sum_{i=1}^{n}\sum_{s=1}^{d} (c\,\vec{x}_i^{(s)} - \vec{y}_i^{(s)})\,\vec{x}_i^{(s)}.$$

Setting this to zero gives

$$\sum_{i=1}^{n}\sum_{s=1}^{d} (c\,\vec{x}_i^{(s)} - \vec{y}_i^{(s)})\,\vec{x}_i^{(s)} = 0 \implies c\sum_{i=1}^{n}\sum_{s=1}^{d} (\vec{x}_i^{(s)})^2 = \sum_{i=1}^{n}\sum_{s=1}^{d} \vec{x}_i^{(s)}\,\vec{y}_i^{(s)},$$

Compare this minimizer to the one we saw in Eq. (1).

so

$$c^* = \frac{\sum_{i=1}^{n}\sum_{s=1}^{d} \vec{x}_i^{(s)}\,\vec{y}_i^{(s)}}{\sum_{i=1}^{n}\sum_{s=1}^{d} (\vec{x}_i^{(s)})^2} = \frac{\sum_{i=1}^{n} \vec{x}_i^{\top}\vec{y}_i}{\sum_{i=1}^{n} \|\vec{x}_i\|^2} \tag{5}$$

Here, we use the transpose notation $\vec{x}_i^{\top}\vec{y}_i$ to replace the dot product expression $\sum_{s=1}^{d} \vec{x}_i^{(s)}\,\vec{y}_i^{(s)}$ which appears in the numerator. We check the second derivative:

$$\frac{\mathrm{d}^2R}{\mathrm{d}c^2} = \frac{2}{n}\sum_{i=1}^{n}\sum_{s=1}^{d} (\vec{x}_i^{(s)})^2,$$

which is strictly positive whenever the data is not entirely equal to the zero vector. By the second derivative test, this critical point is a global minimizer.

Example 1.3.4 *Meteorological modeling*

Morning wind vectors $\vec{x}_i$, indicating both direction and speed, blow across a coastal research station. The researchers want to predict the afternoon wind vec-

tors $\vec{y}_i$ under the observed hypothesis

$$f(c;\, \vec{x}) = c\,\vec{x}, \qquad c \in \mathbb{R}.$$

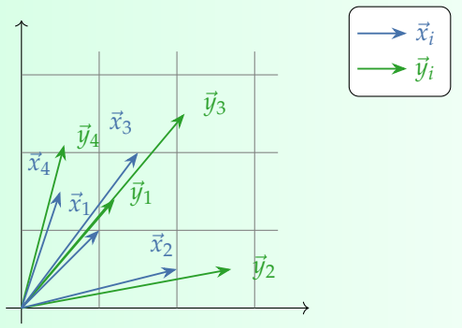
In other words, the afternoon winds tend to blow in the same direction as the morning winds but with a different magnitude. Data was collected over four days, yielding the following measurements:

$\vec{x}_i$	$\vec{y}_i$	Comment
$(1, 1)$	$(1.20,\ 1.40)$	light breeze
$(2, 0.5)$	$(2.70,\ 0.50)$	gusty burst
$(1.5, 2)$	$(2.10,\ 2.50)$	seabreeze mix
$(0.5, 1.5)$	$(0.55,\ 2.10)$	swirl

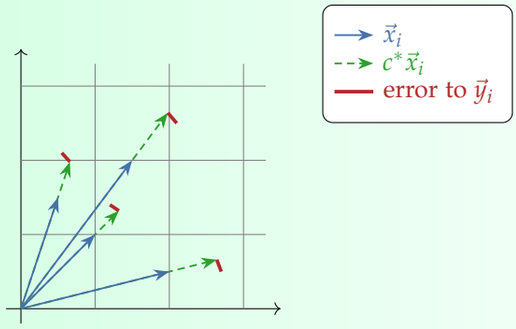
Using the formula derived earlier,

$$c^* \;=\; \frac{\sum_i \vec{x}_i^{\top}\, \vec{y}_i}{\sum_i \|\vec{x}_i\|^2} \;\approx\; \frac{19.825}{15} \;\approx\; 1.32.$$

The following figure depicts the morning wind vectors $\vec{x}_i$ in blue and the observed afternoon wind vectors $\vec{y}_i$ in green:



Finally, we compare the predicted afternoon vectors $c^*\vec{x}_i$ (dashed green arrows) against the actual observations (solid green arrows), with the red lines indicating the prediction error for each day:



One thing worth emphasizing is that in the preceding examples, although each of the features and targets were vectors, the model weights consisted only of a single parameter $c \in \mathbb{R}$; so that even though we are learning a model with variables in high dimensions, the tools needed to find the corresponding minimizer do not go beyond those which

we have seen so far and which are borrowed from single-variable calculus. In the next chapter, this will change.

1.4 Exercises

Exercise 1.1

Determine whether each statement is true or false. Explain your answers.

(a) An input variable is another name for a target.

(b) Binary variables can only take on two values, such as 0 or 1.

(c) A loss function measures the difference between predicted output and actual target.

(d) Numerical variables must be integers only.

Exercise 1.2

Fill in the blank with the appropriate term:

(a) A set of parameters in a _____ are often referred to as _____.

(b) If a variable can be any real number, it is called a _____ variable.

(c) The difference between a model's prediction and the true target is measured by a _____.

Exercise 1.3

Give a brief explanation in response to each question below based on the material presented in the chapter. There can be several correct answers.

(a) Why do we distinguish between training data and testing data?

(b) What is the primary purpose of a risk function in modeling?

(c) In the context of the modeling method, which step ensures we have a way to measure how well or poorly our model is performing, and why?

Exercise 1.4

You measure the temperature (in degrees Celsius) of a beaker of water in a freezer at five different times (measured in hours). Answer each of the following questions. Explain your reasoning and calculations.

(a) What might you wish to model in this situation? Determine the input and output variables for this scenario.

(b) Which of the following models is most appropriate in this situation?

$$f(c;\, x) = ce^{-x} \qquad g(c;\, x) = cx^2 \qquad h(c;\, x) = c \ln x$$

(c) Based on your answer to the previous part, write down the formulas for the associated square loss and empirical risk in terms of $x_1, x_2, \ldots, x_5$ and $y_1, y_2, \ldots, y_5$.

(d) Find the minimizer c^* for the empirical risk in the previous part.

(e) Suppose you collect the following data:

x_i	1	2	3	4	5
y_i	22.22	10.23	4.11	1.25	0.25

Using the formula you found in the previous part for c^*, plug in the data you collected and write down your final model. *(Python may be useful here.)*

(f) If you collect one final data point $(6, 0.15)$ and use it as a test case, what is the value of the squared loss with respect to your trained model?

Exercise 1.5

You track the population of a bacterial culture at six different time points (in hours) and record the total count of bacteria (in thousands). Answer each of the following questions. Explain your reasoning and calculations.

(a) What might you wish to model in this situation? Determine the input and output variables for this scenario.

(b) Which of the following models is most appropriate in this situation? You may assume that food and resources are available in unlimited quantities to the bacterial population.

$$f(c;\, x) = \tfrac{c}{x} \qquad g(c;\, x) = c\sqrt{x} \qquad h(c;\, x) = ce^{x}$$

(c) Based on your answer to the previous part, write down the formulas for the associated square loss and empirical risk in terms of $x_1, x_2, \ldots, x_6$ and $y_1, y_2, \ldots, y_6$.

(d) Find the minimizer c^* for the empirical risk in the previous part.

(e) Suppose you collect the following data:

x_i	1	2	3	4	5	6
y_i	5.4	14.8	40.0	109.2	290.1	815.3

Using the formula you found in the previous part for c^*, plug in the data you collected and write down your final model. *(Python may be useful here.)*

(f) If you collect one final data point $(7, 2456.7)$ and use it as a test case, what is the value of the squared loss with respect to your trained model?

Exercise 1.6

You have collected the following data about a certain plant species' height y_i (measured in centimeters) at six weeks of growth after using different amounts of fertilizer x_i (measured in grams) in the water:

x_i	1.1	2.4	3.2	4.6	5.5
y_i	2.0	3.8	5.0	5.9	6.4

You suspect the relationship follows the model $f(c; x) = c\ln(x)$. Complete the following tasks in a `Jupyter` notebook, and explain your steps:

(a) Define a `Python` function `model(x, c)` that returns $c \ln x$.

(b) Write a `Python` function `risk(c)` that computes the empirical risk associated to the p-loss for $p = 3$ and a given value of $c \in \mathbb{R}$.

(c) Define an array which contains values of c which range from 0 to 100, using $n = 1,000$ sample points. Then evaluate `risk(c)` at each point and store the results in a separate array. Determine the value of c^* in the array which leads to minimal risk.

(d) Produce a scatterplot of the points (x_i, y_i) and overlay the line of best fit $f(c; x) = c^* \ln x$. Be sure to:

- Include a main title;
- Label the horizontal and vertical axes appropriately;
- Show a legend.

If you find yourself getting stuck on some of the visualization steps, the `matplotlib` documentation may be helpful: https://matplotlib.org/stable/index.html.

Exercise 1.7

Give a brief explanation in response to each question below based on the material presented in the chapter. There can be several correct answers.

(a) Define supervised learning and unsupervised learning in your own words.

(b) Which type of learning would best describe each of the following scenarios?

(i) Discerning whether the migration patterns of two herds of wild flamingos intersect;

(ii) Predicting how many pounds of carrots will be sold at a local grocery store based on previous years' data;

(iii) Using readings from a heart monitor to predict cardiac arrest.

Exercise 1.8

Determine whether each statement is true or false. Explain your answers.

(a) The constant model is an example of supervised learning.

(b) The minimizer of the empirical risk for the square loss is always unique.

(c) The mean of a dataset is always one of the data points.

(d) The absolute loss is more sensitive to outliers than the square loss.

(e) The constant model always outputs different values for different inputs.

Exercise 1.9

Tom's bakery is well-known for its sourdough and was recently featured in a local magazine. Tom records the number of customers each day in the first week after the story appears: $\{15, 25, 28, 36, 27\}$. Using the constant model:

(a) Compute the minimizer of the empirical risk for the square loss.

(b) Compute the minimizer of the empirical risk for the absolute loss.

(c) If an outlier of 93 customers show up on the following day, how do the minimizers change?

Exercise 1.10

Complete the following tasks in a `Jupyter` notebook, and explain your steps:

(a) Generate $n = 100$ numbers uniformly at random from the interval $[a, b]$, where a, b are fixed parameters, and store them in an array `x`.

(b) Define two functions `mean(x)` and `median(x)` which return their respective quantities (if the median is not unique, simply return one of the possible medians). Apply these functions to the data generated in (a).

(c) Repeat this process $N = 1000$ times, and store each corresponding mean and median in a separate array.

(d) Plot two histograms: one of the means and one of the medians. Plot them on the same figure.

(e) How do they compare? Explain your findings. It may help to play around with different choices of a, b.

Exercise 1.11

Let $\{x_1, x_2, \ldots, x_n\}$ be a given dataset of real numbers and let $\overline{x}$ denote its mean. Suppose we add a new data point x_{n+1} and produce a new mean $\overline{x}'$ in the process. Find a formula for $\overline{x}'$ in terms of $\overline{x}$.

Exercise 1.12

Determine whether each statement is true or false. If the statement is true, write a proof. If it is false, provide a counterexample.

(a) If x^* is a minimizer for $f(x)$, then x^* is a minimizer for $f(x)^2$.

(b) If x^* is a maximizer for $f(x)$ and $f(x) > 0$ for all x, then x^* is a maximizer for $\ln f(x)$.

(c) If x^* is a minimizer for $f(x)$, and $g : \mathbb{R} \to \mathbb{R}$ is a monotone increasing function then x^* is a maximizer for $g(f(x))$.

(d) If x^* is a maximizer for $f(x)$, and $g : \mathbb{R} \to \mathbb{R}$ is a monotone decreasing function then x^* is a minimizer for $g(f(x))$.

Exercise 1.13

Consider the constant model for real-valued scalar input data. For a given loss function, define the *breakdown point* p to be the smallest proportion of data that, when modified at will, can cause the minimizer of the empirical risk to diverge to infinity.

(a) Show that the breakdown point of the square loss is $1/n$. That is, if $\{x_1, x_2, \ldots, x_{n-1}\}$ are any *fixed* data points, and $x_n = y$ for arbitrary $y \in \mathbb{R}$, prove $\lim_{y\to\infty} \overline{x} = \infty$.

(b) Assume the data $\{x_1, x_2, \ldots, x_n\}$ are all distinct and n is odd, i.e., the median is unique. Show that the breakdown point of the absolute loss is 0.5. The proof consists of two steps:

(i) Suppose you modify strictly less than half of the data by defining $x'_{\lceil n/2\rceil+1}, \ldots, x'_n = y$ for $y \in \mathbb{R}$. Show that

$$\lim_{y\to\infty} \text{median}\{x_1, x_2, \ldots, x'_{\lceil n/2\rceil+1}, \ldots, x'_n\} = \max\{x_1, x_2, \ldots, x_{\lceil n/2\rceil}\}$$

In other words, the minimizer of the empirical risk does not diverge to infinity.

(ii) Suppose you modify strictly greater than half of the data by defining $x'_{\lceil n/2\rceil}, \ldots, x'_n = y$ for $y \in \mathbb{R}$. Show that

$$\lim_{y\to\infty} \text{median}\{x_1, x_2, \ldots, x'_{\lceil n/2\rceil}, \ldots, x'_n\} = \infty.$$

(c) Revisiting the **Key Idea** at the end of Section 1.2, explain how you can interpret (a) and (b) in the context of sensitivity to outliers.

Exercise 1.14

The *Huber loss function* is a hybrid loss function that behaves quadratically for small deviations and linearly for large deviations. It is defined as follows for a

given dataset $\{x_1, x_2, \ldots, x_n\}$:

$$L_{\text{Huber}}(c; x_i) = \begin{cases} \frac{1}{2}(c - x_i)^2, & \text{if } |c - x_i| \leq \frac{1}{2}, \\ \frac{1}{2}|c - x_i| - \frac{1}{8}, & \text{if } |c - x_i| > \frac{1}{2}. \end{cases}$$

The empirical risk function for a given choice of c then becomes:

$$R_{\text{Huber}}(c) = \frac{1}{n} \sum_{i=1}^{n} L_{\text{Huber}}(c; x_i).$$

(a) Compute $\frac{\mathrm{d}}{\mathrm{d}c} R_{\text{Huber}}(c)$ explicitly. Clearly state where it is differentiable and explain why the function is not differentiable at certain points.

(b) Assume your data x_i have no large deviations. In particular, assume all datapoints lie in a small interval $(-\delta, \delta)$ around $x = 0$ for $\delta < 1/4$. Prove that the minimizer for $R_{\text{Huber}}(c)$ is $c^* = \overline{x}$.

(c) Explain why the minimizer for $R_{\text{Huber}}(c)$ should be closer to the median of the data if your data has large deviations and is very spread out on a large interval, then. Feel free use examples or `Python` plots.

Exercise 1.15

Let $\{x_i\}_{i=1}^n$ be a collection of n real numbers, and recall the square loss for the constant model $f(c;\, x) = c$:

$$L_{\text{sq}}(c;\, x_i) = (c - x_i)^2.$$

In this problem we will work with the **maximum risk**, which is given by:

$$R_{\text{max}}(c) = \max_{1 \leq i \leq n} L_{\text{sq}}\big(c;\, x_i\big).$$

(a) Show that

$$R_{\text{max}}(c) \;=\; \big(\max_{1 \leq i \leq n} |c - x_i|\,\big)^2.$$

(b) Assume the data is ordered from least to greatest: $x_1 \leq x_2 \leq \cdots \leq x_n$. Show that if c^* is a minimizer of $R_{\text{max}}(c)$, then c^* lies in the closed interval

$$[\, x_1,\, x_n\,].$$

Hint: Assume that c^* did not lie in this interval; can it still be a minimizer?

(c) Show that the **midrange**, given by

$$c^* \;=\; \frac{x_1 + x_n}{2},$$

is *a* minimiser of $R_{\text{max}}(c)$ and that it is *unique* when $x_{(1)} \neq x_{(n)}$.

(d) Explain how sensitive c^* is to outliers when compared to the mean and the median of the dataset (you may use examples here).

Exercise 1.16

Determine whether each statement is true or false. Explain your answers.

(a) The feature dimension d represents the number of data points in a dataset.

(b) The target dimension p refers to the number of features used to represent a data point.

(c) The vector-valued constant model assigns the same scalar value to each coordinate of the output vector.

(d) The vector-valued simplified linear model can be used when the feature and target dimensions are different.

(e) The empirical risk function for the vector-valued constant model is minimized by the component-wise median of the dataset.

Exercise 1.17

Suppose all feature vectors in a dataset $\{\vec{x}_i\}_{i=1}^n$ are scaled by a factor $\alpha > 0$, so that each $\vec{x}_i$ is replaced with $\alpha\vec{x}_i$. Explain how the minimizer c^* of the vector-valued constant model changes under this transformation.

Exercise 1.18

Throughout the book we use summation notation very often. Work through the following without a calculator.

(a) Prove that $\sum_{i=1}^{n}(x_i - \overline{x}) = 0$ directly from the definition of $\overline{x}$.

(b) Evaluate the telescoping sum

$$S_n \;=\; \sum_{k=1}^{n}\big[(k+1)^2 - k^2\big], \qquad n \geq 1.$$

Your answer will depend on n.

(c) Let $\{a_{ij}\}_{1\leq i\leq n, 1\leq j\leq m}$ be real numbers. Show that

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} \;=\; \sum_{j=1}^{m}\sum_{i=1}^{n} a_{ij}.$$

(d) Look up formulas for each of the following expressions:

$$\sum_{i=1}^{n} i$$
$$\sum_{i=1}^{n} i^2$$
$$\sum_{i=1}^{n} cr^i \quad c, r > 0.$$

You do not need to write proofs but you are encouraged to review them.

Exercise 1.19

Fix $d \geq 1$. Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^d$ be fixed vectors and let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times d}$ be fixed square matrices. Recall our convention for the dot-product notation:

$$\vec{u}^\top \vec{v} = \sum_{s=1}^{d} u^{(s)} v^{(s)}.$$

Here, $(\cdot)^\top$ denotes matrix transpose.

(a) Prove both of the identities below:

(i) $\vec{u}^\top (\vec{v} + \vec{w}) = \vec{u}^\top \vec{v} + \vec{u}^\top \vec{w}$,

(ii) $(\mathbf{A}\vec{u})^\top \vec{v} = \vec{u}^\top \mathbf{A}^\top \vec{v}$.

You may use the summation form of matrix multiplication or properties of matrix transpose.

(b) With $d = 3$, let

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 3 & 4 \\ 0 & 2 & 5 \end{bmatrix}, \quad \vec{u} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 4 \\ 0 \\ -1 \end{bmatrix}.$$

Compute:

(i) $\mathbf{A}\vec{u}$,

(ii) $\vec{u}^\top \vec{v}$,

(iii) $\mathbf{A}^\top \mathbf{A}$,

(iv) $\frac{\vec{u}^\top \vec{v}}{\|\vec{u}\|^2}$.

Exercise 1.20

A coastal research station records hourly surface current velocities

$$\vec{x}_i = (\vec{x}_i^{(1)}, \vec{x}_i^{(2)}) \in \mathbb{R}^2,$$

measured in metres per second, where $\vec{x}_i^{(1)}$ is the eastward component and $\vec{x}_i^{(2)}$ is the northward component. Oceanographers suspect that during the study period the motion is driven predominantly by a single tidal stream that always points in the direction

$$\vec{d} = \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

but whose *speed* (magnitude) is unknown. They therefore propose the vector-valued constant model

$$f(c;\, \vec{x}) \;=\; c\,\vec{d}, \qquad c \in \mathbb{R}.$$

Six velocity measurements (in $\mathrm{m\,s^{-1}}$) are listed below:

$$\{\vec{x}_i\}_{i=1}^{6} = \Big\{(0.60, 1.21),\ (0.70, 1.47),\ (0.50, 0.93),\ (0.65, 1.25),\ (0.55, 1.11),\ (0.72, 1.38)\Big\}.$$

(a) Identify the input and output variables, state the model, and write down a formula for the square loss $L_{\mathrm{sq}}(c;\, \vec{x})$ in this setting. Is this an unsupervised or supervised scenario?

(b) Write the empirical risk function

$$R(c) = \frac{1}{6} \sum_{i=1}^{6} L_{\mathrm{sq}}(c;\, \vec{x}_i)$$

for the data above and derive the critical-point equation

$$c^{*} = \frac{\displaystyle\sum_{i=1}^{n} \vec{d}^{\top} \vec{x}_i}{\displaystyle\sum_{i=1}^{n} \|\vec{d}\|^2}.$$

(c) Verify that the second derivative is positive and conclude that c^{*} is the unique global minimiser.

(d) Using a few lines of `Python`, evaluate the sums in part (b) for the six observations and report the numerical value of c^{*}.

(e) Interpret $c^{*}\vec{d}$ physically. How well does this single vector summarise the observed currents? Comment on the suitability of the constant-direction assumption.

Exercise 1.21

A portable air-quality device measures two pollutant concentrations (ozone and nitrogen dioxide) simultaneously. Because the factory calibration drifts over time, a laboratory performs reference measurements $\vec{y}_i = (y_i^{O_3}, y_i^{NO_2})^\top$ on the same air samples for which the field device reports $\vec{x}_i = (x_i^{O_3}, x_i^{NO_2})^\top$. Engineers hypothesise that both channels are off by a common multiplicative factor $c \in \mathbb{R}$; that is,

$$f(c; \vec{x}_i) = c\,\vec{x}_i,$$

which is realized as the vector-valued simplified linear model. Five paired readings (all in ppb) are given below:

i	$\vec{x}_i$	$\vec{y}_i$
1	(9.0, 12.0)	(27.3, 36.6)
2	(10.5, 14.2)	(30.8, 41.4)
3	(8.3, 11.1)	(24.7, 33.6)
4	(11.2, 15.0)	(33.7, 45.5)
5	(9.7, 13.0)	(28.6, 38.4)

Work through the modeling recipe.

(a) Clearly state the input variables, output variables, model, square loss function, and empirical risk function (use $n = 5$ and $d = p = 2$).

(b) Using the results in this chapter and a bit of `Python` code where necessary, find the value c^* which minimizes the empirical risk associated to the square loss. You do not need to derive the formula(s) from scratch.

(c) Explain why c^* must be positive for any non-zero dataset.

(d) Suppose future field readings are multiplied by c^*. What improvement (or risk) does this bring when monitoring pollutant levels in real time?

2 The Linear Model

"All models are wrong, but some are useful."

- *George Box*

Linear regression lies at the very heart of statistical learning: by fitting a straight line (or more generally, linear subspace) through data, we can attempt to capture the simplest relationship between features and targets, in the process laying the groundwork for more elaborate models. This chapter will systematically cover the linear model from the "simple" case of scalar features and targets to the most general case of vector features and targets.

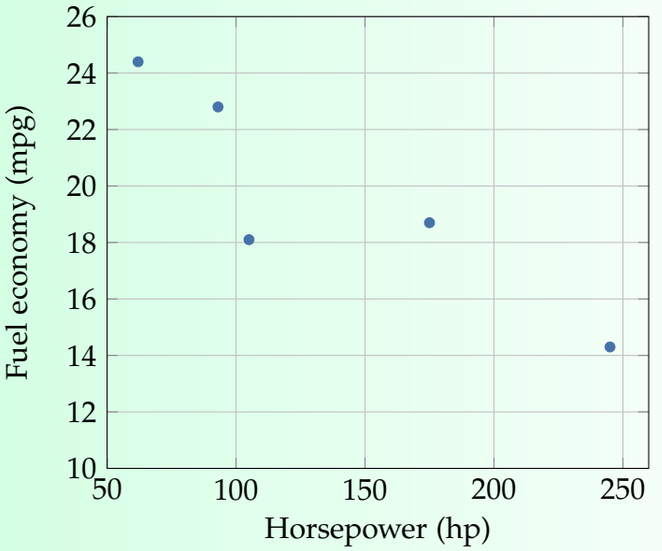
2.1 Simple Linear Regression: Scalar Features, Scalar Targets

Example 2.1.1 ***Predicting fuel economy from horsepower***

To motivate our discussions of the linear model, consider the relationship between engine power and fuel efficiency using five cars from the Motor Trend "mtcars" dataset (1974)[a]. In each case we have a single feature x_i given by the horsepower (hp) of the engine, and a single target y_i given by the observed fuel economy in miles per gallon (mpg):

Model	Datsun 710	Merc 240D	Valiant	Hornet Sportabout	Duster 360
Horsepower x_i (hp)	93	62	105	175	245
Fuel economy y_i (mpg)	22.8	24.4	18.1	18.7	14.3

By visualizing the points from the dataset on a two-dimensional set of axes, one observes a downward-sloping trend: as horsepower increases, mpg tends to decrease.



Thus we might attempt to model the relationship using a line of the form

$$y = a\,x + b$$

to quantify exactly "how many mpg you should expect given the vehicle horsepower." The modeling method gives us a framework to identify the best slope and intercept for the scenarios like this.

[a] See https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/mtcars

Simple linear regression is one of the most foundational statistical models, and serves as a building block for more advanced machine learning methods. The **simple linear model** is used to model a scalar-valued target y in terms of a scalar-valued feature x using a linear function with trainable weights $a, b \in \mathbb{R}$. Formally, given an input-output pair (x, y), the model we consider takes the form:

$$f(a, b;\, x) = ax + b,$$

where $a, b \in \mathbb{R}$ are the weights of the model that we seek to learn from data. The parameter a is often called the **slope** of the model, and the parameter b is often called the **intercept** or bias of the model. **Simple linear regression** refers to the process of finding the best parameters a^*, b^* for a specific collection of training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. The line $y = a^*x + b^*$ is sometimes called the **line of best fit** for the training data.

The single biggest difference between the simple linear model and the models we considered in Chapter 1, and what will become the main source of complexity for the sections to follow, is the fact that the model f contains *more than one* trainable parameter; in this case two: $a, b \in \mathbb{R}$. However, the process of training the linear model remains generally the same overall. To quantify the error associated with a particular choice of a, b and a fixed input-output pair (x, y), we can use the square loss:

$$L_{\text{sq}}(a, b; (x, y)) = (y - (ax + b))^2. \tag{6}$$

The empirical risk is then given by the mean loss over all data points:

$$R(a, b; \{(x_i, y_i)\}_{i=1}^n) = \frac{1}{n} \sum_{i=1}^n (y_i - (ax_i + b))^2. \tag{7}$$

As we did in Chapter 1, we may write $R(a, b)$ for short, with the understanding that $R(a, b)$ is a function of a, b determined by the data points $\{(x_i, y_i)\}_{i=1}^n$. Now since $R(a, b)$ is a differentiable function of two variables, we can find its minimizer by searching for critical points. First we compute its partial derivatives:

$$\begin{aligned}
\frac{\partial R}{\partial a} &= \frac{\partial}{\partial a} \frac{1}{n} \sum_{i=1}^n \big(y_i - (ax_i + b)\big)^2 \\
&= \frac{1}{n} \sum_{i=1}^n 2\big(y_i - (ax_i + b)\big) \frac{\partial}{\partial a} \big(y_i - (ax_i + b)\big) \\
&= \frac{2}{n} \sum_{i=1}^n \big(y_i - (ax_i + b)\big) (-x_i) \\
&= -\frac{2}{n} \sum_{i=1}^n x_i \Big(y_i - (ax_i + b)\Big); \\
\frac{\partial R}{\partial b} &= \frac{\partial}{\partial b} \frac{1}{n} \sum_{i=1}^n \big(y_i - (ax_i + b)\big)^2 \\
&= \frac{1}{n} \sum_{i=1}^n 2\big(y_i - (ax_i + b)\big) \frac{\partial}{\partial b} \big(y_i - (ax_i + b)\big) \\
&= \frac{2}{n} \sum_{i=1}^n \big(y_i - (ax_i + b)\big) (-1) \\
&= -\frac{2}{n} \sum_{i=1}^n \Big(y_i - (ax_i + b)\Big).
\end{aligned}$$

which are then set equal to zero:

$$\begin{aligned}
-\frac{2}{n} \sum_{i=1}^n x_i \Big(y_i - (a^* x_i + b^*)\Big) &= 0, \\
-\frac{2}{n} \sum_{i=1}^n \Big(y_i - (a^* x_i + b^*)\Big) &= 0.
\end{aligned} \tag{8}$$

The pair of equations given in Eq. (8) are examples of **normal equations**, which generally speaking are the equations formed by setting the partial derivatives of the empirical risk associated to a particular model and loss function equal to zero; and which depend on the parameters of the model in addition to the training data. Multiplying both equations by $-\frac{n}{2}$ yields

$$\begin{aligned}
\sum_{i=1}^n x_i \Big(y_i - (a^* x_i + b^*)\Big) &= 0, \\
\sum_{i=1}^n \Big(y_i - (a^* x_i + b^*)\Big) &= 0.
\end{aligned}$$

The second equation can be rearranged into the form

$$\sum_{i=1}^{n} y_i = a^* \sum_{i=1}^{n} x_i + nb^*,$$

so that

$$b^* = \frac{1}{n}\left(\sum_{i=1}^{n} y_i - a^* \sum_{i=1}^{n} x_i\right) = \overline{y} - a^*\,\overline{x}.$$

Next, substituting $b^* = \overline{y} - a^*\,\overline{x}$ into the first equation gives

$$\sum_{i=1}^{n} x_i\Big(y_i - \Big(a^* x_i + \overline{y} - a^*\,\overline{x}\Big)\Big) = 0.$$

Expanding the expression inside the summation leads to

$$\sum_{i=1}^{n} x_i\Big(y_i - \overline{y}\Big) - a^* \sum_{i=1}^{n} x_i\Big(x_i - \overline{x}\Big) = 0.$$

Solving for a^* yields the well-known formula for the slope:

$$a^* = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

Once a^* is determined, the optimal intercept is given by

$$b^* = \overline{y} - a^*\,\overline{x}.$$

To further verify that the parameters (a^*, b^*) indeed minimize the empirical risk $R(a,b)$, we apply the second derivative test for functions of two variables (see Theorem A.3). The Hessian matrix of R can be found by calculating the three second-order partial derivatives of R as follows:

$$\begin{aligned}
\frac{\partial^2}{\partial a^2} R(a,b) &= \frac{\partial}{\partial a}\left[-\frac{2}{n}\sum_{i=1}^{n} x_i\Big(y_i - (ax_i + b)\Big)\right] = \frac{2}{n}\sum_{i=1}^{n} x_i^2 \\
\frac{\partial^2}{\partial b \partial a} R(a,b) &= \frac{\partial}{\partial b}\left[-\frac{2}{n}\sum_{i=1}^{n} x_i\Big(y_i - (ax_i + b)\Big)\right] = \frac{2}{n}\sum_{i=1}^{n} x_i \\
\frac{\partial^2}{\partial b^2} R(a,b) &= \frac{\partial}{\partial b}\left[-\frac{2}{n}\sum_{i=1}^{n}\Big(y_i - (ax_i + b)\Big)\right] = 2.
\end{aligned}$$

Therefore the Hessian is given by:

$$H(a,b) = \begin{pmatrix} \frac{2}{n}\sum_{i=1}^{n} x_i^2 & \frac{2}{n}\sum_{i=1}^{n} x_i \\ \frac{2}{n}\sum_{i=1}^{n} x_i & 2 \end{pmatrix}.$$

? Why does $H(a,b)$ not depend on a, b at all? (The reason is related to the fact that the loss function is *quadratic*)

In this case, it happens that $H(a,b)$ does *not* depend on any specific point a, b; rather, it depends only on the training data $\{(x_i, y_i)\}_{i=1}^{n}$. The upper-left element $\frac{\partial^2}{\partial a^2} R(a,b)$ is positive, and the determinant of the Hessian is

$$\det(H) = \frac{4}{n^2}\left(n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right),$$

which is positive as long as all of the x_i are not identical. This can be seen as follows. Let $\vec{x} \in \mathbb{R}^n$ denote the vector containing the training features x_i, and let $\vec{\mathbf{1}} \in \mathbb{R}^n$ denote the vector of all ones. By the Cauchy-Schwarz inequality (see Theorem B.3), we have

$$\left(\sum_{i=1}^{n} x_i\right)^2 = |\vec{x}^\top \vec{\mathbf{1}}|^2 \leq \|\vec{x}\|^2 \|\vec{\mathbf{1}}\|^2 = n\left(\sum_{i=1}^{n} x_i^2\right),$$

and unless $\vec{x}$ and $\vec{\mathbf{1}}$ are linearly dependent (which happens only when all of the x_i's are constant), the inequality will be *strict*, that is

$$n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2 > 0.$$

Therefore, the critical point (a^*, b^*) is a global minimum of $R(a, b)$. These parameters define the best-fit line under the square loss. We summarize this discussion in the form of a theorem below.

Theorem 2.1.2

Let $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ denote a collection of training data consisting of scalar-valued features x_i and targets y_i. Suppose we wish to model a target y in terms of each input feature x using a linear function $f(a, b;\, x)$ given by the formula

$$f(a, b;\, x) = ax + b.$$

Then the optimal parameters $a^*, b^* \in \mathbb{R}$ which minimize the empirical risk $R(a, b)$ (see Eq. (7)) for the squared loss (see Eq. (6)) are given by

$$\begin{aligned} a^* &= \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \\ b^* &= \overline{y} - a^*\,\overline{x}. \end{aligned} \tag{9}$$

Example 2.1.3 *Predicting sales from advertising*

Loretta owns a small business and records her weekly advertising expenditure (in thousands of dollars) and the corresponding sales (in thousands of dollars). Suppose the data is

x_i (advertising)	1.16	2.24	2.88	4.1	5.1
y_i (sales)	2.16	2.81	3.44	4.72	6.31

The sample means are $\overline{x} = 3.096$ and $\overline{y} = 3.888$. We can compute

$$\sum_{i=1}^{5}(x_i - \overline{x})(y_i - \overline{y}) \approx 10.054$$

$$\sum_{i=1}^{5}(x_i - \overline{x})^2 \approx 9.551.$$

Thus, based on Eq. (9), the optimal slope is given by

$$a^* = \frac{\sum_{i=1}^{5}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{5}(x_i - \overline{x})^2} \approx \frac{10.054}{9.551} \approx 1.053,$$

and similarly the intercept is

$$b^* = \overline{y} - a^* \overline{x} = (3.888) - a^*(3.096) \approx 0.629.$$

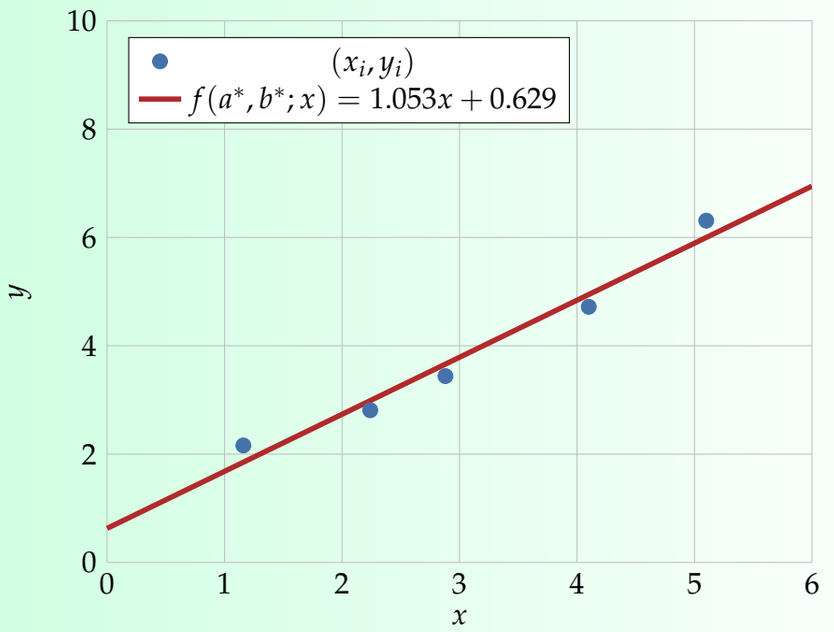
The resulting model, i.e., line of best-fit for Loretta's data, is given by

$$f(a^*, b^*; x) = 1.053x + 0.629.$$

The following `Python` code demonstrates the computation and visualization of this model:

```
import numpy as np
import matplotlib.pyplot as plt

# Data: Advertising expenditure (in $1000) and Sales (in $1000)
x = np.array([1.16, 2.24, 2.88, 4.1 , 5.1 ])
y = np.array([2.16, 2.81, 3.44, 4.72, 6.31])

# Compute means
x_mean = np.mean(x)
y_mean = np.mean(y)

# Compute optimal slope and intercept
a = np.sum((x - x_mean) * (y - y_mean)) / np.sum((x - x_mean)**2)
b = y_mean - a * x_mean

print("Optimal model: f(x) = {:.2f}x + {:.2f}".format(a, b))

# Plot data and fitted line
plt.scatter(x, y, color='blue', label='Data')
x_fit = np.linspace(min(x), max(x), 100)
plt.plot(x_fit, a*x_fit + b, color='red', label='Fitted line')
plt.xlabel('Advertising (in $1000)')
plt.ylabel('Sales (in $1000)')
plt.title('Simple Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```

The scatterplot and line of best fit generated by the `Python` code above should roughly resemble the picture below.



We can push the theoretical setup established in this chapter a bit further and see, for example, how the formulas we derived hold up when we want to make small changes to our training data. Suppose for example we have a dataset $\{(x_i, y_i)\}_{i=1}^n$ where the values x_i are measured in years. For example, we might have $x_1 = 2025$ and $x_2 = 2020$. The units are somewhat large and unwieldy, so what if instead we considered features x_i' which are given by "years since 2020," in other words, by setting $x_i' = x_i - 2020$? The following example follows in the footsteps of Example 1.2.5 and addresses this question.

Example 2.1.4 *Effect of Translating Features*

Consider a simple linear regression model trained on data $\{(x_i, y_i)\}_{i=1}^n$. Suppose we translate every feature by some fixed constant $\alpha \in \mathbb{R}$, i.e., define new features

$$x_i' = x_i + \alpha.$$

We can use the formulas provided in Theorem 2.1.2 on the translated data to see how the optimal parameters and line of best fit change. First, the new mean is

$$\overline{x'} = \overline{x} + \alpha.$$

Notice that

$$x_i' - \overline{x'} = (x_i + \alpha) - (\overline{x} + \alpha) = x_i - \overline{x},$$

so that the computation for the new slope, which we might call a', remains unchanged:

$$a' = \frac{\sum_{i=1}^n (x_i' - \overline{x'})(y_i - \overline{y})}{\sum_{i=1}^n (x_i' - \overline{x'})^2} = \frac{\sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^n (x_i - \overline{x})^2} = a^*.$$

The new intercept, denoted b', becomes

$$b' = \overline{y} - a^* \overline{x'} = \overline{y} - a^*(\overline{x} + \alpha) = (\overline{y} - a^* \overline{x}) - a^* \alpha = b^* - a^* \alpha.$$

Thus, translating all training features by a constant α leaves the slope unchanged while shifting the intercept by $-a^*\alpha$. Concretely, our new parameters a', b' satisfy

$$a' = a^*$$
$$b' = b^* - a^*\alpha.$$

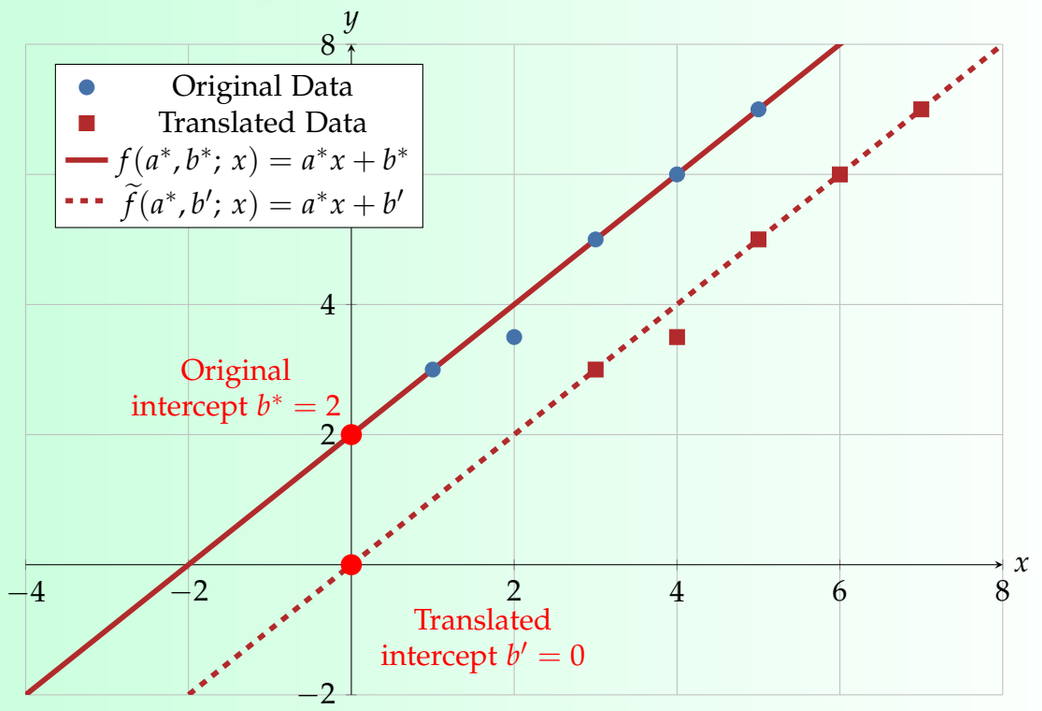
For example, if the original model is given by

$$f(a^*, b^*; x) = 1.6x + 0.2,$$

translating the features by $\alpha = 2$ produces a new model $\widetilde{f}$ given by

$$\widetilde{f}(a', b'; x) = 1.6x + \Big(0.2 - 1.6 \times 2\Big) = 1.6x - 3.0.$$

Below, we include an illustration of this example. The original best-fit line $y = f(a^*, b^*; x)$ intercepts the y-axis at $(0, b^*)$, while the translated model $\widetilde{f}(a', b'; x)$ (dashed red) intercepts at the new location $(0, b')$.



There are many other ways we could explore how the optimal parameters for a simple linear regression model behave under transformations of our data; the preceding example serves merely to give us a sense of how we would approach such a question. We continue this theme in the exercises at the end of this chapter.

A quick note on terminology: the *empirical risk associated with the square loss* comes up quite a bit in data science, so we usually refer to it by the alternative name **mean squared error (MSE)**. More specifically, we define

$$\text{MSE}(\vec{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \vec{w}^\top \vec{x}_i + b)^2. \tag{10}$$

2.2 Warmup: The constant model revisited

In Section 2.1 we stepped up the complexity of our toolkit by considering models of the form $f(a, b; x) = ax + b$ which contain two parameters. As we go forward into the remainder of the chapter, we will only be expanding on this trend by scaling up from two parameters to models which contain as many as we want. In this warmup section, we will take a moment to return to the constant model we introduced in Section 1.2, and later in Example 1.3.1. This time, we assume our training data are vector-valued, i.e. $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$. The **vector-valued constant model** is defined by

$$f(\vec{w}; \vec{x}) = \vec{w},$$

where $\vec{w} \in \mathbb{R}^d$ is a single vector containing d entries. In this unsupervised scenario, the goal is to find the vector $\vec{w}$ which minimizes a given risk function and thus best "summarizes" our data. It is important to note that although f has a very simple structure, the vector $\vec{w}$ contains d weights, so the resulting optimization problem is going up a step in complexity even if the model itself is relatively simple. To measure the error incurred by a particular choice of $\vec{w}$, we use the square loss:

$$L_{\text{sq}}(\vec{w}; \vec{x}_i) = \|\vec{w} - \vec{x}_i\|^2 = \sum_{j=1}^{d} (\vec{w}^{(j)} - \vec{x}_i^{(j)})^2. \tag{11}$$

The empirical risk, as usual, is then given by the average loss over the dataset:

$$R(\vec{w}; \{\vec{x}_i\}_{i=1}^n) = \frac{1}{n} \sum_{i=1}^{n} L_{\text{sq}}(\vec{w}; \vec{x}_i) = \frac{1}{n} \sum_{i=1}^{n} \|\vec{w} - \vec{x}_i\|^2. \tag{12}$$

To find the minimizer $\vec{w}^*$ we need to compute the gradient of $R(\vec{w})$ with respect to $\vec{w}$ and set it equal to the zero vector. We can do this by viewing R as a function of d variables $\vec{w}^{(1)}, \vec{w}^{(2)}, \ldots, \vec{w}^{(d)} \in \mathbb{R}$ and computing, for each $1 \leq j \leq d$, the partial derivative $\frac{\partial}{\partial \vec{w}^{(j)}} R$. Fixing j and using Eq. (11), we have

$$\begin{aligned}
\frac{\partial}{\partial \vec{w}^{(j)}} R(\vec{w}) &= \frac{\partial}{\partial \vec{w}^{(j)}} \frac{1}{n} \sum_{i=1}^{n} \|\vec{w} - \vec{x}_i\|^2 \\
&= \frac{\partial}{\partial \vec{w}^{(j)}} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} (\vec{w}^{(k)} - \vec{x}_i^{(k)})^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial}{\partial \vec{w}^{(j)}} \sum_{k=1}^{d} (\vec{w}^{(k)} - \vec{x}_i^{(k)})^2 \right]
\end{aligned}$$

Next we investigate closely the expression in the brackets on the right:

$$\frac{\partial}{\partial \vec{w}^{(j)}} \sum_{k=1}^{d} (\vec{w}^{(k)} - \vec{x}_i^{(k)})^2 = \frac{\partial}{\partial \vec{w}^{(j)}} \left[(\vec{w}^{(1)} - \vec{x}_i^{(1)})^2 + (\vec{w}^{(2)} - \vec{x}_i^{(2)})^2 + \ldots + (\vec{w}^{(d)} - \vec{x}_i^{(d)})^2 \right].$$

Notice that since j is fixed, the term in the sum will only depend on $\vec{w}^{(j)}$ whenever the index k matches j; otherwise the derivative is zero. So when we evaluate the partial derivative, most of the terms vanish and we have:

$$\frac{\partial}{\partial \vec{w}^{(j)}} \sum_{k=1}^{d} (\vec{w}^{(k)} - \vec{x}_i^{(k)})^2 = \frac{\partial}{\partial \vec{w}^{(j)}} (\vec{w}^{(j)} - \vec{x}_i^{(j)})^2 = 2(\vec{w}^{(j)} - \vec{x}_i^{(j)})$$

Therefore, returning to the preceding calculation, we have

$$\begin{aligned}\frac{\partial}{\partial \vec{w}^{(j)}} R(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n}\left[2(\vec{w}^{(j)} - \vec{x}_i^{(j)})\right] \\ &= 2\vec{w}^{(j)} - \frac{2}{n}\sum_{i=1}^{n}\vec{x}_i^{(j)}.\end{aligned}$$

Notice that $\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i^{(j)}$ is actually the j-th coordinate of the mean *vector* $\vec{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i$. Thus

$$\frac{\partial}{\partial \vec{w}^{(j)}} R(\vec{w}) = 2(\vec{w}^{(j)} - \vec{\bar{x}}^{(j)}).$$

Since this expression holds for any given coordinate index j, we can summarize our findings by writing the following description of $\nabla R(\vec{w})$:

$$\nabla R(\vec{w}) = \begin{bmatrix} \frac{\partial}{\partial \vec{w}^{(1)}} R(\vec{w}) \\ \frac{\partial}{\partial \vec{w}^{(2)}} R(\vec{w}) \\ \vdots \\ \frac{\partial}{\partial \vec{w}^{(d)}} R(\vec{w}) \end{bmatrix} = \begin{bmatrix} 2(\vec{w}^{(1)} - \vec{\bar{x}}^{(1)}) \\ 2(\vec{w}^{(2)} - \vec{\bar{x}}^{(2)}) \\ \vdots \\ 2(\vec{w}^{(d)} - \vec{\bar{x}}^{(d)}) \end{bmatrix} = 2(\vec{w} - \vec{\bar{x}}).$$

Therefore, after dividing by the factor of two, we can conclude that $\nabla R(\vec{w}^*) = \vec{0}$ if and only if $\vec{w}^* = \vec{\bar{x}}$.

Verifying that the critical point we have found is a true global minimum requires some more advanced linear algebra and multivariable calculus (see Theorem A.4). In DSC 40A, if the model under consideration has one or two parameters you must verify that a given critical point is a true minimizer of the empirical risk using the second derivative test; if it has more than two parameters, you are allowed and encouraged to skip this step.

We can write this conclusion as a theorem below.

Theorem 2.2.1 ***Optimal Constant Model for Vector Features***

Let $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$ denote a collection of training data consisting of vector-valued features. Suppose we model the $\vec{x}_i$'s using a single vector given $\vec{w}$ using the formula

$$f(\vec{w}; \vec{x}) = \vec{w}.$$

Then the unique minimizer of the empirical risk $R(\vec{w}; \{\vec{x}_i\}_{i=1}^n)$ (see Eq. (12)) associated to the square loss $L_{\text{sq}}(\vec{w}; \vec{x}_i)$ (see Eq. (11)) is given by the vector mean of the training data:

$$\vec{w}^* = \vec{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i.$$

Example 2.2.2 *Network traffic baseline for anomaly detection*

A corporate information security firm wishes to design a model to detect anomalous network behavior. Each network flow observation is represented by a 2-dimensional feature vector $\vec{x}_i$ with components given by:

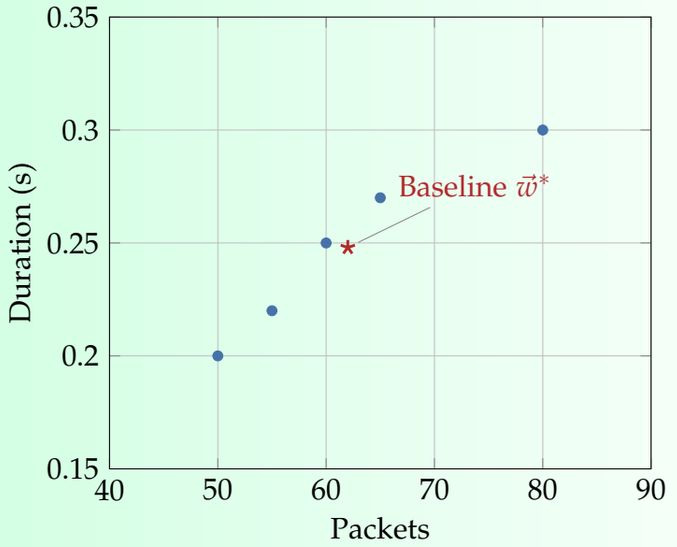
$\vec{x}_i^{(1)}$	packet count
$\vec{x}_i^{(2)}$	flow duration

The vector-valued constant model selects the single "baseline" flow signature $\vec{w}^*$ that minimizes $\frac{1}{n}\sum_i \|\vec{w} - \vec{x}_i\|^2$, i.e. the mean flow vector. New traffic instances whose flow signature vectors are far from $\vec{w}^*$ can then be flagged as anomalies. Suppose we observe five flows:

Flow i	Packet count	Flow duration (s)
$\vec{x}_1$	50	0.20
$\vec{x}_2$	60	0.25
$\vec{x}_3$	55	0.22
$\vec{x}_4$	80	0.30
$\vec{x}_5$	65	0.27

The mean vector is

$$\vec{w}^* = \left(\frac{50+60+55+80+65}{5},\ \frac{0.20+0.25+0.22+0.30+0.27}{5}\right) \approx (62,\ 0.248).$$



This "star" marks the constant-model baseline. In practice, any new flow whose feature vector falls far from this point (in Euclidean distance) is treated as anomalous.

2.3 Multiple Linear Regression: Vector Features, Scalar Targets

Now that we have warmed up our multivariable optimization skills, we can turn

the page to the next topic on our journey, which is multiple regression. Multiple linear regression *extends* the idea of simple linear regression to the case where each training example consists of a feature vector $\vec{x} \in \mathbb{R}^d$ and a scalar target $y \in \mathbb{R}$. The main idea remains the same: we will attempt to model y as a linear function of $\vec{x}$; however, the number of parameters and structure of the function will be different than before.

Some scenarios where a multiple linear regression model might be appropriate include, e.g., predicting housing prices from various attributes, estimating sales based on multiple indicators, or forecasting outcomes in medical studies where several measurements are taken. The **multiple linear regression model** is given by the general formula

$$f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b, \tag{13}$$

where $\vec{w} = \begin{bmatrix} \vec{w}^{(1)} & \vec{w}^{(2)} & \dots & \vec{w}^{(d)} \end{bmatrix}^\top \in \mathbb{R}^d$ is a vector of weights and $b \in \mathbb{R}$ is the intercept parameter. More explicitly, if we write

$$\vec{x} = \begin{bmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \\ \vdots \\ \vec{x}^{(d)} \end{bmatrix}, \quad \vec{w} = \begin{bmatrix} \vec{w}^{(1)} \\ \vec{w}^{(2)} \\ \vdots \\ \vec{w}^{(d)} \end{bmatrix},$$

then Eq. (13) can be expanded into the form

$$\begin{aligned} f(\vec{w}, b; \vec{x}) &= \begin{bmatrix} \vec{w}^{(1)} & \vec{w}^{(2)} & \dots & \vec{w}^{(d)} \end{bmatrix} \begin{bmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \\ \vdots \\ \vec{x}^{(d)} \end{bmatrix} + b \\ &= \vec{w}^{(1)}\vec{x}^{(1)} + \vec{w}^{(2)}\vec{x}^{(2)} + \ldots + \vec{w}^{(d)}\vec{x}^{(d)} + b. \end{aligned}$$

In other words...

Key Idea

The multiple linear regression model $f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b$ can be thought of as taking each feature coordinate $\vec{x}^{(j)}$, applying a corresponding weight $\vec{w}^{(j)}$ which can be positive or negative, adding up each term, and finally adding an "offset" or "intercept" b.

Before we tackle the general overarching question of how to find the optimal parameters $\vec{w}^*$ and b^* for the multiple linear regression model, let's first consider an example in dimension $d = 2$.

Example 2.3.1

Rosabel works at a real estate firm and wants to design a model to predict the final sales price (in thousands) of a home given two datapoints which can be found in her database: square feet of the house, and the mean sales price (in thousands) for all homes in the same ZIP code during the last year. Her dataset consists of three houses which have recently sold along with their closing sales prices, given below.

square feet	$\vec{x}_i^{(1)}$	1285	2250	2125
mean sales price, ZIP code in thousands	$\vec{x}_i^{(2)}$	158	99	124
actual sales price in thousands	y_i	325	221	445

In the language of input-output or feature-target pairs, Rosabel's training dataset can also be described as the set

$$\left\{ \left(\begin{bmatrix} 1285 \\ 158 \end{bmatrix}, 325 \right), \left(\begin{bmatrix} 2250 \\ 99 \end{bmatrix}, 221 \right), \left(\begin{bmatrix} 2125 \\ 124 \end{bmatrix}, 445 \right) \right\}.$$

Rosabel decides to use a multiple linear regression model as follows. She defines the model $f(\vec{w}, b; \vec{x})$ by the general formula

$$f\left(\vec{w}, b; \begin{bmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \end{bmatrix}\right) = \vec{w}^{(1)}\vec{x}^{(1)} + \vec{w}^{(2)}\vec{x}^{(2)} + b,$$

where the numbers $\vec{w}^{(1)}, \vec{w}^{(2)}, b \in \mathbb{R}$ are to be determined. Her next goal is to find which values of these parameters lead to the most accurate model f. To quantify the error of a prediction for a single training pair $(\vec{x}, y)$, she decides to use the square loss

$$L_{\text{sq}}(\vec{w}, b; (\vec{x}, y)) = (y - (\vec{w}^\top \vec{x} + b))^2 = \Big(y - \big(\vec{w}^{(1)}\, \vec{x}^{(1)} + \vec{w}^{(2)}\, \vec{x}^{(2)} + b\big)\Big)^2.$$

The empirical risk over Rosabel's dataset is then given by

$$R(\vec{w}, b) = \frac{1}{3}\sum_{i=1}^{3}\Big(y_i - \big(\vec{w}^{(1)}\, \vec{x}_i^{(1)} + \vec{w}^{(2)}\, \vec{x}_i^{(2)} + b\big)\Big)^2.$$

To find the parameters that minimize R, we set the partial derivatives with respect to $\vec{w}^{(1)}$, $\vec{w}^{(2)}$, and b to zero. This will require a bit of work, so let's take things one step at a time. First, we have

$$\begin{aligned} \frac{\partial}{\partial \vec{w}^{(1)}} R(\vec{w}, b) &= -\frac{2}{3}\sum_{i=1}^{3} \vec{x}_i^{(1)}\Big(y_i - \big(\vec{w}^{(1)}\, \vec{x}_i^{(1)} + \vec{w}^{(2)}\, \vec{x}_i^{(2)} + b\big)\Big) \\ &= -\frac{2}{3}\begin{bmatrix} \vec{x}_1^{(1)} \\ \vec{x}_2^{(1)} \\ \vec{x}_3^{(1)} \end{bmatrix}^\top \begin{bmatrix} y_1 - (\vec{w}^\top \vec{x}_1 + b) \\ y_2 - (\vec{w}^\top \vec{x}_2 + b) \\ y_3 - (\vec{w}^\top \vec{x}_3 + b) \end{bmatrix} \\ &= -\frac{2}{3}\begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \begin{bmatrix} 325 - (\vec{w}^\top \vec{x}_1 + b) \\ 221 - (\vec{w}^\top \vec{x}_2 + b) \\ 445 - (\vec{w}^\top \vec{x}_3 + b) \end{bmatrix} \end{aligned} \tag{14}$$

where in the second line we re-wrote the sum in the first line as a dot product over the entries of two vectors obtained by inspecting the original expression. The third line arises by replacing the generic notation $\vec{x}_i^{(j)}$ and y_i with the actual values from Rosabel's dataset. It turns out that this equation is a bit more complicated

than the ones we have seen so far; so we can attempt to simplify things a bit in a step-by-step manner. Let's begin by defining a vector

$$\vec{\theta} = \begin{bmatrix} b \\ \vec{w}^{(1)} \\ \vec{w}^{(2)} \end{bmatrix},$$

which will contain all of our model parameters simultaneously. In a similar manner, let's go ahead and define a modified version of each feature vector by adding a one in the first component. Specifically, we write

$$\vec{z}_1 = \begin{bmatrix} 1 \\ 1285 \\ 158 \end{bmatrix}, \vec{z}_2 = \begin{bmatrix} 1 \\ 2250 \\ 99 \end{bmatrix}, \vec{z}_3 = \begin{bmatrix} 1 \\ 2125 \\ 124 \end{bmatrix}.$$

These vectors $\vec{z}_1, \vec{z}_2, \vec{z}_3$ are called the **design vectors**, or **augmented feature vectors**, of the dataset and are simply used to rewrite the normal equations for the multiple regression model in a less horrendous manner. Now let's return to Eq. (14). Using $\vec{\theta}, \vec{z}_1, \vec{z}_2, \vec{z}_3$ instead, we can write things in the form

$$\begin{aligned} \frac{\partial}{\partial \vec{w}^{(1)}} R(\vec{w}, b) &= -\frac{2}{3} \begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \begin{bmatrix} 325 - (\vec{w}^\top \vec{x}_1 + b) \\ 221 - (\vec{w}^\top \vec{x}_2 + b) \\ 445 - (\vec{w}^\top \vec{x}_3 + b) \end{bmatrix} \\ &= -\frac{2}{3} \begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \left(\begin{bmatrix} 325 \\ 221 \\ 445 \end{bmatrix} - \begin{bmatrix} \vec{z}_1^\top \vec{\theta} \\ \vec{z}_2^\top \vec{\theta} \\ \vec{z}_3^\top \vec{\theta} \end{bmatrix} \right) \end{aligned} \tag{15}$$

Next we can recognize that the right-hand side can be realized as a matrix-vector product, since each coordinate is a dot product of the same vector $\vec{\theta}$ with various vectors $\vec{z}_1$. That is, if we define the matrix $\mathbf{Z} \in \mathbb{R}^{3\times 3}$ given by

$$\mathbf{Z} = \begin{bmatrix} \vec{z}_1^\top \\ \vec{z}_2^\top \\ \vec{z}_3^\top \end{bmatrix} = \begin{bmatrix} 1 & 1285 & 158 \\ 1 & 2250 & 99 \\ 1 & 2125 & 124 \end{bmatrix},$$

then Eq. (15) becomes

$$-\frac{2}{3} \begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \left(\begin{bmatrix} 325 \\ 221 \\ 445 \end{bmatrix} - \begin{bmatrix} \vec{\theta}^\top \vec{z}_1 \\ \vec{\theta}^\top \vec{z}_2 \\ \vec{\theta}^\top \vec{z}_3 \end{bmatrix} \right) = -\frac{2}{3} \begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \left(\vec{y} - \mathbf{Z}\vec{\theta} \right).$$

The matrix $\mathbf{Z}$ is called the **design matrix**. Now all of this arises from only $\frac{\partial}{\partial \vec{w}^{(1)}} R(\vec{w}, b)$. If we repeat these steps for $\frac{\partial}{\partial \vec{w}^{(2)}} R(\vec{w}, b)$ and $\frac{\partial}{\partial b} R(\vec{w}, b)$, then we have

You should attempt to repeat these steps on your own.

the following family of three normal equations

$$\frac{\partial}{\partial \vec{w}^{(1)}} R(\vec{w}, b) = -\frac{2}{3} \begin{bmatrix} 1285 \\ 2250 \\ 2125 \end{bmatrix}^\top \left(\vec{y} - \mathbf{Z}\,\vec{\theta} \right) = 0$$

$$\frac{\partial}{\partial \vec{w}^{(2)}} R(\vec{w}, b) = -\frac{2}{3} \begin{bmatrix} 158 \\ 99 \\ 124 \end{bmatrix}^\top \left(\vec{y} - \mathbf{Z}\,\vec{\theta} \right) = 0$$

$$\frac{\partial}{\partial b} R(\vec{w}, b) = -\frac{2}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^\top \left(\vec{y} - \mathbf{Z}\,\vec{\theta} \right) = 0.$$

Each of the numerical column vectors which appear in the right-hand side are actually just columns of the design matrix $\mathbf{Z}$. Using what we know about matrix-vector multiplication, the entire system of normal equations can therefore be recast into a single matrix-vector equation given by

$$\nabla R(\vec{w}, b) = -\frac{2}{3} \mathbf{Z}^\top \left(\vec{y} - \mathbf{Z}\,\vec{\theta} \right) = \vec{0}.$$

This is the payoff of our investment into reformatting the training data: *the normal equations, which initially are quite complicated, can be expressed as a more streamlined matrix-vector equation with the unknown* $\vec{\theta}$. The normal equations can then be rearranged into the form

$$\mathbf{Z}^\top \mathbf{Z}\,\vec{\theta} = \mathbf{Z}^\top \vec{y}.$$

Thus, if $(\mathbf{Z}^\top \mathbf{Z})^{-1}$ exists, the minimizer is unique and is given by

$$\vec{\theta} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \vec{y}.$$

A straightforward numerical computation using `Python` yields the approximate matrices

$$\mathbf{Z}^\top \mathbf{Z} \approx \begin{bmatrix} 3 & 5660 & 381 \\ 5660 & 11229350 & 689280 \\ 381 & 689280 & 50141 \end{bmatrix},$$

$$(\mathbf{Z}^\top \mathbf{Z})^{-1} \approx \begin{bmatrix} 313.455 & -0.075 & -1.344 \\ -0.075 & 0.000 & 0.000 \\ -1.344 & 0.000 & 0.006 \end{bmatrix},$$

$$\mathbf{Z}^\top \vec{y} \approx \begin{bmatrix} 991 \\ 1860500 \\ 128409 \end{bmatrix}$$

Therefore, we have that

$$\vec{\theta}^* = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \vec{y} \approx \begin{bmatrix} -2405.796 \\ 0.634 \\ 12.129 \end{bmatrix},$$

so that the estimated intercept is $b^* \approx -2405.796$, and the weight coefficients are $\vec{w}^{(1)*} \approx 0.634$ and $\vec{w}^{(2)*} \approx 12.129$. Consequently, Rosabel's multiple linear regression model takes the form

$$f\left(\vec{\theta}^*; \begin{bmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \end{bmatrix}\right) \approx 0.634\,\vec{x}^{(1)} + 12.129\,\vec{x}^{(2)} - 2405.796.$$

We can verify the model's predictions by computing matrix vector product $\mathbf{Z}\,\theta$, which should hopefully be close to $\vec{y}$. As it turns out, we get exactly that:

$$\mathbf{Z}\,\theta^* = \begin{bmatrix} 325 \\ 221 \\ 445 \end{bmatrix}.$$

After exploring Rosabel's example with feature dimension $d = 2$, we are now ready to turn to the general case. We will use many of the same steps that we saw in Example 2.3.1. Before we begin, let's summarize a couple of the important points we just encountered.

Design Vectors and Matrices

Suppose we have a collection of training data $\{(\vec{x}_i, y_i)\}_{i=1}^n$ where $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ and wish to model y using a multiple linear regression model $f(\vec{w}, b; \vec{x})$ of the form

$$f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b.$$

An important preliminary step when setting up this problem is to construct design vectors, which are the vectors $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_n \in \mathbb{R}^{d+1}$ obtained by adding a leading one in the first entry:

$$\vec{z}_i = \begin{bmatrix} 1 \\ \vec{x}_i \end{bmatrix}.$$

The design matrix $\mathbf{Z} \in \mathbb{R}^{n \times (d+1)}$ is simply the matrix with rows given by $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_n \in \mathbb{R}^{d+1}$. That is,

$$\mathbf{Z} = \begin{bmatrix} \vec{z}_1^\top \\ \vec{z}_2^\top \\ \vdots \\ \vec{z}_n^\top \end{bmatrix} = \begin{bmatrix} 1 & \vec{x}_1^\top \\ 1 & \vec{x}_2^\top \\ \vdots & \\ 1 & \vec{x}_n^\top \end{bmatrix}$$

Now we will state our main result concerning the optimal weights $\vec{w}^*, b$ as a theorem and prove it carefully.

Theorem 2.3.2 *Optimal Model Parameters for Multiple Linear Regression*

Suppose we have a collection of training data $\{(\vec{x}_i, y_i)\}_{i=1}^n$ where $\vec{x}_i \in \mathbb{R}^d$ and

$y_i \in \mathbb{R}$ and model y using a multiple linear regression model $f(\vec{w}, b; \vec{x})$ of the form

$$f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b.$$

Let $\mathbf{Z}$ denote the design matrix associated to the training data and let $\vec{y} \in \mathbb{R}^n$ denote the vector of labels. Assume that $(\mathbf{Z}^\top \mathbf{Z})^{-1}$ exists. Then the parameters $\vec{w}^* \in \mathbb{R}^d$ and $b^* \in \mathbb{R}$ which minimize the mean squared error (see Eq. (10)) associated to the training data are unique and given by the formula

$$\begin{bmatrix} b^* \\ \vec{w}^* \end{bmatrix} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \vec{y}.$$

Proof

For a single training example $(\vec{x}, y)$, the square loss is given by

$$L_{\text{sq}}(\vec{w}, b; (\vec{x}, y)) = \big(y - (\vec{w}^\top \vec{x} + b)\big)^2.$$

Thus, the empirical risk (or MSE) is then the mean loss over all n training examples:

$$R(\vec{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - (\vec{w}^\top \vec{x}_i + b)\big)^2.$$

Since $R(\vec{w}, b)$ is differentiable with respect to $\vec{w}$ and b, we find its minimizer by setting its partial derivatives equal to zero. We will use the shorthand $\vec{\theta} \in \mathbb{R}^{d+1}$ to capture all of the parameters in one vector by setting $\vec{\theta} = \begin{bmatrix} b & \vec{w}^{(1)} & \vec{w}^{(2)} & \dots & \vec{w}^{(d)} \end{bmatrix}^\top$. First, differentiating with respect to b we obtain

$$\begin{aligned} \frac{\partial R}{\partial b} &= -\frac{2}{n} \sum_{i=1}^{n} \Big(y_i - (\vec{w}^\top \vec{x}_i + b)\Big) \\ &= -\frac{2}{n} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}^\top \begin{bmatrix} y_1 - \vec{z}_1^\top \vec{\theta} \\ y_2 - \vec{z}_2^\top \vec{\theta} \\ \vdots \\ y_n - \vec{z}_n^\top \vec{\theta} \end{bmatrix} = -\frac{2}{n} \vec{\mathbf{1}}_n^\top \left(\vec{y} - \mathbf{Z}\vec{\theta}\right) \end{aligned} \tag{16}$$

Next, differentiating $R(\vec{w}, b)$ with respect to $\vec{w}^{(j)}$ for some $1 \leq j \leq d$ fixed, we have

$$\begin{aligned} \frac{\partial R}{\partial \vec{w}^{(j)}} &= -\frac{2}{n} \sum_{i=1}^{n} \vec{x}_i^{(j)} \Big(y_i - (\vec{w}^\top \vec{x}_i + b)\Big) \\ &= -\frac{2}{n} \begin{bmatrix} \vec{x}_1^{(j)} \\ \vec{x}_2^{(j)} \\ \vdots \\ \vec{x}_n^{(j)} \end{bmatrix}^\top \begin{bmatrix} y_1 - \vec{z}_1^\top \vec{\theta} \\ y_2 - \vec{z}_2^\top \vec{\theta} \\ \vdots \\ y_n - \vec{z}_n^\top \vec{\theta} \end{bmatrix} = -\frac{2}{n} \begin{bmatrix} \vec{x}_1^{(j)} \\ \vec{x}_2^{(j)} \\ \vdots \\ \vec{x}_n^{(j)} \end{bmatrix}^\top \left(\vec{y} - \mathbf{Z}\vec{\theta}\right). \end{aligned} \tag{17}$$

Therefore, for the same reasons as we saw in Example 2.3.1, we can re-write Eqs. (16) to (17) in the more compact form

$$\nabla R(\vec{\theta}) = -\frac{2}{n} \mathbf{Z}^\top (\vec{y} - \mathbf{Z}\,\vec{\theta}).$$

So the normal equations are given by

$$\mathbf{Z}^\top (\vec{y} - \mathbf{Z}\vec{\theta}^*) = \vec{0} \iff \mathbf{Z}^\top \mathbf{Z}\vec{\theta}^* = \mathbf{Z}^\top \vec{y}.$$

Since we assumed that $\mathbf{Z}^\top \mathbf{Z}$ is invertible, the equation $(\mathbf{Z}^\top \mathbf{Z})\vec{\theta}^* = \mathbf{Z}^\top \vec{y}$ has a unique solution given by

$$\vec{\theta}^* = \begin{bmatrix} b^* \\ \vec{w}^* \end{bmatrix} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \vec{y}.$$

To conclude this section we provide an example demonstrating how to implement a multiple linear regression model in practice.

🍊 Example 2.3.3

In economics, analysts often wish to forecast key indicators such as retail sales based on several factors. Consider an economist Zihan who wants to predict monthly retail sales (in billions USD) for a city using three features: average household income (in thousands), unemployment rate (in percent), and consumer confidence index (scale 0–100). Zihan's dataset below comprises 10 observations from different cities.

mean income	$\vec{x}_i^{(1)}$	50	60	55	65	48	70	52	68	58	63
unemployment rate	$\vec{x}_i^{(2)}$	5	4	6	3	7	2	6	3	5	4
consumer confidence index	$\vec{x}_i^{(3)}$	80	85	75	90	70	95	78	88	80	82
retail sales	y_i	120	150	130	160	110	180	125	165	140	155

He decides to model the relationship using multiple linear regression with hypothesis function:

$$f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b,$$

where $\vec{x} \in \mathbb{R}^3$ contains the three features and $f(\vec{w}, b; \vec{x})$ predicts sales. The empirical risk, or mean squared error (MSE), is given by

$$R(\vec{w}, b) = \frac{1}{10} \sum_{i=1}^{10} \Big(y_i - (\vec{w}^\top \vec{x}_i + b) \Big)^2.$$

To solve for the optimal parameters, we first construct the design matrix $\mathbf{Z} \in \mathbb{R}^{10 \times 4}$ by appending a column of ones to the feature matrix. Then the solution to the normal equations can be written in compact form as

$$\begin{bmatrix} b^* \\ \vec{w}^* \end{bmatrix} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \vec{y}.$$

The following Python code demonstrates how to compute the optimal parameters, make predictions, and evaluate the model by comparing predicted targets with actual targets and computing the MSE.

```
import numpy as np

# Dataset: [Income, Unemployment, Confidence]
X_data = np.array([
    [50, 5, 80],
    [60, 4, 85],
    [55, 6, 75],
    [65, 3, 90],
    [48, 7, 70],
    [70, 2, 95],
    [52, 6, 78],
    [68, 3, 88],
    [58, 5, 80],
    [63, 4, 82]
])
# Target: Retail Sales (in billions)
y = np.array([120, 150, 130, 160, 110, 180, 125, 165, 140, 155])

# Number of data points
n = X_data.shape[0]

# Construct design matrix by adding a column of ones
X = np.hstack((np.ones((n, 1)), X_data))

# Compute optimal parameters using normal equations
theta = np.linalg.inv(X.T @ X) @ (X.T @ y)
b_opt = theta[0]
w_opt = theta[1:]

print("Optimal parameters:")
print("Intercept (b):", b_opt)
print("Weights (w):", w_opt)

# Compute predictions
y_pred = X @ theta

# Compute Mean Squared Error (MSE)
mse = np.mean((y - y_pred)**2)

print("\nPredictions and Errors:")
for i in range(n):
    print("Input:", X_data[i], "Actual:", y[i],
            "Predicted:", round(y_pred[i],2))
print("\nMean Squared Error (MSE):", round(mse,2))
```

Running the code, Zihan obtains the following outputs.

```
Optimal parameters:
```

```
Intercept (b): -64.77167424862273
Weights (w): [2.41276322 0.90903936 0.75418279]

Predictions and Errors:
Input: [50  5 80] Actual: 120 Predicted: 120.75
Input: [60  4 85] Actual: 150 Predicted: 147.74
Input: [55  6 75] Actual: 130 Predicted: 129.95
Input: [65  3 90] Actual: 160 Predicted: 162.66
Input: [48  7 70] Actual: 110 Predicted: 110.2
Input: [70  2 95] Actual: 180 Predicted: 177.59
Input: [52  6 78] Actual: 125 Predicted: 124.97
Input: [68  3 88] Actual: 165 Predicted: 168.39
Input: [58  5 80] Actual: 140 Predicted: 140.05
Input: [63  4 82] Actual: 155 Predicted: 152.71

Mean Squared Error (MSE): 3.54
```

2.4 The General Linear Model: Vector Features, Vector Targets

In this section, we bring all of the topics and techniques in this chapter to their natural conclusion and wrap up our discussion of linear models. First and foremost, we note that many real-world problems require us to predict not just a single target, but an entire vector of targets. In Example 2.3.3, for instance, Zihan may very well have set out to forecast multiple interrelated indicators simultaneously. For example, suppose Zihan was interested in predicting *both* the total retail sales *and* the inflation rate for a city based on its consumer confidence, unemployment rate, and average household income. In this setting, the target is a vector $y \in \mathbb{R}^p$ (with $p = 2$ in our example) while the features remain in $\mathbb{R}^d$ (with $d = 3$). In this setting, Zihan might choose to use **general linear model** instead of multiple regression; this model has the form

$$f(\mathbf{W}, \vec{b}; \vec{x}) = \mathbf{W}\vec{x} + \vec{b},$$

where $\mathbf{W} \in \mathbb{R}^{p \times d}$ is a weight *matrix* and $\vec{b} \in \mathbb{R}^p$ is an **intercept (or bias) vector**. In this setting, we are given a collection of vector-valued training examples of the form

$$\{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \ldots, (\vec{x}_n, \vec{y}_n)\},$$

then the mean squared error can be defined in a similar manner as in earlier sections:

$$R(\mathbf{W}, \vec{b}) = R(\mathbf{W}, \vec{b}; \{(\vec{x}_i, \vec{y}_i)\}_{i=1}^n) = \frac{1}{n} \sum_{i=1}^{n} \|\vec{y}_i - (\mathbf{W}\vec{x}_i + \vec{b})\|^2. \tag{18}$$

The main result we wish to cover for this section is given as a theorem below. The proof of this result requires some careful calculation along the lines of the proof of Theorem 2.3.2, but is otherwise not fundamentally different. For this reason, we omit a proof.

The reader is encouraged to attempt a proof of this result on their own.

Theorem 2.4.1 *Optimal Model Parameters for General Linear Model*

Suppose we have a collection of training data $\{(\vec{x}_i, \vec{y}_i)\}_{i=1}^n$ where $\vec{x}_i \in \mathbb{R}^d$ and $\vec{y}_i \in \mathbb{R}^p$ for each $1 \leq i \leq n$; suppose we model $\vec{y}$ using a multivariate linear regression model $f(\mathbf{W}, \vec{b}; \vec{x})$ of the form

$$f(\mathbf{W}, \vec{b}; \vec{x}) = \mathbf{W}\vec{x} + \vec{b},$$

where $\mathbf{W} \in \mathbb{R}^{p \times d}$ and $\vec{b} \in \mathbb{R}^p$. Let $\mathbf{Z} \in \mathbb{R}^{n \times (d+1)}$ denote the design matrix associated to the training data and let

$$\mathbf{Y} = \begin{bmatrix} \vec{y}_1^\top \\ \vec{y}_2^\top \\ \vdots \\ \vec{y}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times p}$$

denote the matrix containing all target vectors. Assume that $(\mathbf{Z}^\top \mathbf{Z})^{-1}$ exists. Then the parameters $\mathbf{W}^* \in \mathbb{R}^{p \times d}$ and $\vec{b}^* \in \mathbb{R}^p$ which minimize the mean squared error for the training dataset (see Eq. (18)) are unique and are given by the formula

$$\begin{bmatrix} (\vec{b}^*)^\top \\ (\mathbf{W}^*)^\top \end{bmatrix} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{Y}.$$

Example 2.4.2

Dr. Miranda is a cryptozoologist[a] and often encounters reports of unknown creatures with varying characteristics. She is investigating sightings of a legendary creature reputed to inhabit remote forests. She believes that certain environmental factors may help predict two key attributes of the creature: its estimated length (in meters) and its estimated weight (in kilograms). In her study, the three features recorded for each sighting are: forest density (on a scale from 0 to 100), ambient temperature (in °C), and altitude (in meters). The target is a two-dimensional vector consisting of the creature's estimated length and weight. Dr. Miranda collects a dataset of 8 sightings as shown below.

Forest density	$\vec{x}_i^{(1)}$	70	80	60	90	50	85	65	75
Temperature (°C)	$\vec{x}_i^{(2)}$	15	10	20	12	25	18	22	16
Altitude (m)	$\vec{x}_i^{(3)}$	200	150	250	180	300	220	280	210
Estimated length (m)	$\vec{y}_i^{(1)}$	2.5	3.0	2.0	3.5	1.8	3.2	2.2	2.8
Estimated weight (kg)	$\vec{y}_i^{(2)}$	150	170	140	190	130	180	150	160

Dr. Miranda opts to use the general linear model

$$f(\mathbf{W}, \vec{b}; \vec{x}) = \mathbf{W}\vec{x} + \vec{b},$$

where $\vec{x} \in \mathbb{R}^3$, $\mathbf{W} \in \mathbb{R}^{2 \times 3}$, and $\vec{b} \in \mathbb{R}^2$. Her design matrix $\mathbf{Z}$ and target matrix $\mathbf{Y}$

are given by

$$\mathbf{Z} = \begin{bmatrix} 1 & 70 & 15 & 200 \\ 1 & 80 & 10 & 150 \\ 1 & 60 & 20 & 250 \\ 1 & 90 & 12 & 180 \\ 1 & 50 & 25 & 300 \\ 1 & 85 & 18 & 220 \\ 1 & 65 & 22 & 280 \\ 1 & 75 & 16 & 210 \end{bmatrix} \in \mathbb{R}^{8\times 4}, \quad \mathbf{Y} = \begin{bmatrix} 2.5 & 150 \\ 3.0 & 170 \\ 2.0 & 140 \\ 3.5 & 190 \\ 1.8 & 130 \\ 3.2 & 180 \\ 2.2 & 150 \\ 2.8 & 160 \end{bmatrix} \in \mathbb{R}^{8\times 2}.$$

She uses the following Python code to perform the computation of the optimal parameters, predictions, and MSE.

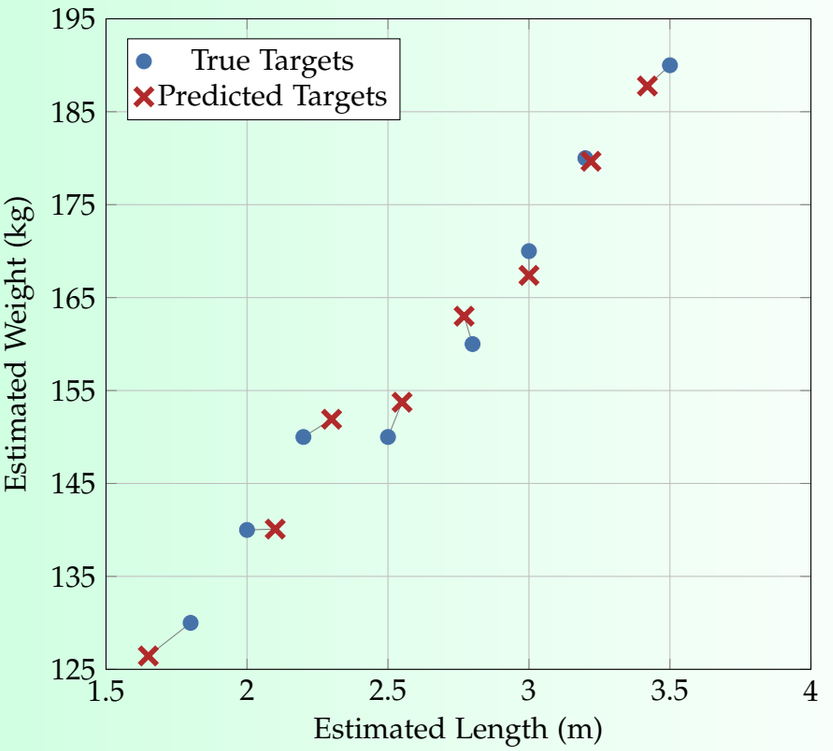
```
import numpy as np

# Dataset: [Forest Density, Temperature, Altitude]
X_data = np.array([
    [70, 15, 200],
    [80, 10, 150],
    [60, 20, 250],
    [90, 12, 180],
    [50, 25, 300],
    [85, 18, 220],
    [65, 22, 280],
    [75, 16, 210]
])
# Targets: [Estimated Length (m), Estimated Weight (kg)]
Y = np.array([
    [2.5, 150],
    [3.0, 170],
    [2.0, 140],
    [3.5, 190],
    [1.8, 130],
    [3.2, 180],
    [2.2, 150],
    [2.8, 160]
])

n = X_data.shape[0]

# Construct design matrix by adding a column of ones
X = np.hstack((np.ones((n, 1)), X_data))  # X in R^(8x4)

# Compute optimal parameters using normal equations
Theta = np.linalg.inv(X.T @ X) @ (X.T @ Y)
b_opt = Theta[0, :]
W_opt = Theta[1:, :].T

print("Optimal parameters:")
```

```
print("Intercept (b):", b_opt)
print("Weights (W):", W_opt)

# Compute predictions
Y_pred = X @ Theta

# Compute Mean Squared Error (MSE)
mse = np.mean(np.sum((Y - Y_pred)**2, axis=1))

print("\nPredictions and Errors:")
for i in range(n):
    print("Input:", X_data[i], "Actual:", Y[i],
          "Predicted:", np.round(Y_pred[i],2))
print("Mean Squared Error (MSE):", np.round(mse,2))
```

Running the code, Dr. Miranda obtains the following outputs:

```
Optimal parameters:
Intercept (b): [-0.4510648  14.08040087]
Weights (W): [[ 4.41942831e-02  1.71324451e+00]
    [ 1.39221045e-02 -1.15818244e+00]
    [-1.52203621e-03  1.85514178e-01]]

Predictions and Errors:
Input: [ 70  15 200] Actual: [  2.5 150. ] Predicted: [  2.55 153.74]
Input: [ 80  10 150] Actual: [  3. 170.] Predicted: [  3.   167.39]
Input: [ 60  20 250] Actual: [  2. 140.] Predicted: [  2.1  140.09]
Input: [ 90  12 180] Actual: [  3.5 190. ] Predicted: [  3.42 187.77]
Input: [ 50  25 300] Actual: [  1.8 130. ] Predicted: [  1.65 126.44]
Input: [ 85  18 220] Actual: [  3.2 180. ] Predicted: [  3.22 179.67]
Input: [ 65  22 280] Actual: [  2.2 150. ] Predicted: [  2.3  151.91]
Input: [ 75  16 210] Actual: [  2.8 160. ] Predicted: [  2.77 163.  ]
Mean Squared Error (MSE): 3.2
```

A scatterplot can help visualize the performance of the model. The plot below shows, in the space of targets, the true target vectors (blue circles) and the predicted target vectors (red crosses) for each observation.



In this example, the predicted target vectors closely match the true targets, as reflected by the low MSE. Note that the MSE can be thought of as the average squared distance between each pair consisting of a blue circle and red cross.

[a] From Wikipedia: *"Cryptozoology is a pseudoscience and subculture that searches for and studies unknown, legendary, or extinct animals whose present existence is disputed or unsubstantiated, particularly those popular in folklore, such as Bigfoot, the Loch Ness Monster, Yeti, the chupacabra, the Jersey Devil, or the Mokele-mbembe. Cryptozoologists refer to these entities as cryptids, a term coined by the subculture."*

2.5 Exercises

Exercise 2.1

Determine whether each statement is true or false. Explain briefly.

(a) If the training features $x_1, \ldots, x_n$ for a simple linear regression problem have mean zero then the intercept of the optimal linear model will equal zero.

(b) The line of best fit is found by minimizing the empirical risk.

(c) The slope of the line of best fit must always be positive.

(d) A simple linear regression model has exactly two parameters.

Exercise 2.2

Match each term (a)–(e) with its correct definition (1)–(5):

(a)	Empirical risk	Equations obtained by setting partial derivatives of empirical risk to zero.	(1)
(b)	Intercept	Average value of the loss function over the training data.	(2)
(c)	Square loss	Parameter indicating the amount that the predicted value changes if the feature increases by one unit.	(3)
(d)	Normal equations	Parameter indicating the predicted value when the feature is zero.	(4)
(e)	Slope	A loss function measuring the squared difference between prediction and actual value.	(5)

Exercise 2.3

A time traveler is interested in modeling the relationship between temporal displacement (years traveled) and energy consumption (gigawatts). His observations are given below.

Years traveled	5	10	15	20	25
Energy consumed	1.21	2.42	3.60	4.79	5.98

Using the empirical risk associated to the square loss, find the optimal linear regression model predicting energy required for a given number of years traveled. *(You do not need to re-derive our formulas from scratch.)*

Exercise 2.4

An alchemist measures gold created (in grams) against hours spent brewing philosopher's potions and obtains the following dataset.

Hours spent brewing	2	3	5	7	8	11
Gold created	0.5	0.75	1.2	1.7	2.1	2.9

(a) Determine the slope and intercept of the linear model which minimizes mean squared error.

(b) Predict the amount of gold the alchemist will create after 10 hours of brewing philosopher's potions.

Exercise 2.5

Write Python code to verify numerically the observation in Example 2.1.4 that translating the features used to train a simple linear regression model by a constant α affects only intercept but not slope. Use the following guidelines:

(i) Generate a synthetic dataset of points $\{(x_i, y_i)\}_{i=1}^{n}$ as follows. First, fix $a, b \in$

$\mathbb{R}$ of your choice. Then, sample 100 random values x_i in an interval $[0, 10]$. Finally, sample some labels y_i by calculating $y_i = a\,x_i + b + (\text{noise}_i)$ where (noise_i) is a very small random number obtained however you like.

(ii) With your setup $\{(x_i, y_i)\}_{i=1}^{n}$, calculate the optimal parameters a^*, b^* (which should be slightly different from a, b due to the noise added).

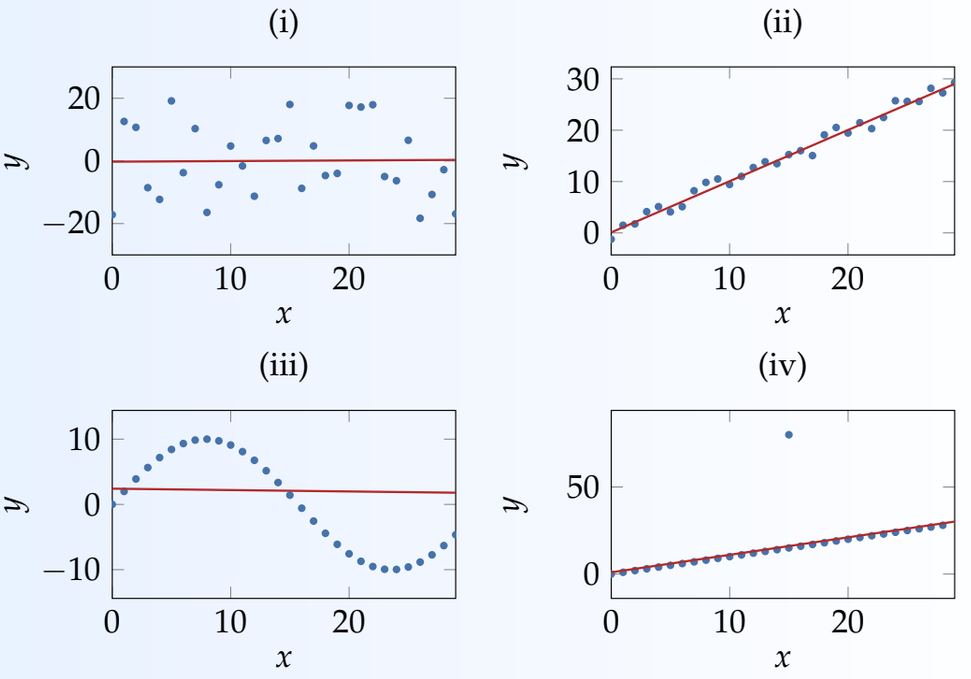
(iii) Next, translate your features x_i by a fixed constant α of your choice, and repeat step (ii) – how do the values change?

Exercise 2.6

The four plots below contain a training dataset $\{(x_i, y_i)\}_{i=1}^{30}$ of thirty points in the xy-plane (blue), along with the line of best fit for the associated simple linear regression problem (red).

(a) For each plot below, determine whether the simple linear model is appropriate. Write a short explanation of your reasoning.

(b) For each plot, consider the value $R(a^*, b^*)$ of the empirical risk associated to the optimal parameters and given training dataset. Rank the plots from least to greatest value of $R(a^*, b^*)$.



Exercise 2.7

Consider a simple linear regression model $f(a^*, b^*; x) = a^*x + b^*$ which has already been trained on data $\{(x_i, y_i)\}_{i=1}^{n}$. Suppose we transform every feature x_i by

a fixed scalar $\beta \in \mathbb{R}$. In other words, we obtain a new collection of features

$$x_i' = \beta x_i.$$

Assuming we do not change the labels y_i, find formulas for the optimal parameters $a', b' \in \mathbb{R}$ of the linear model which minimizes the empirical risk associated to the square loss of the modified training data $\{(x_i', y_i)\}_{i=1}^n$. Your formulas should be in terms of a^*, b^*. (Follow in the footsteps of Example 2.1.4).

Exercise 2.8

In a study of daily ice cream sales, an analyst builds a simple linear regression model to predict sales (in hundreds USD) based on the daily average temperature measured in Celsius. The model is given by

$$f(a^*, b^*; x_C) = a^* x_C + b^*,$$

where x_C is the temperature in Celsius and the optimal parameters found are a^* and b^*. Suppose the analyst decides to convert all temperature measurements from Celsius to Fahrenheit using the transformation

$$x_F = 1.8 x_C + 32.$$

Find formulas for the new optimal parameters a' and b' in terms of a^*, b^* for the model when written in terms of temperature in Fahrenheit, that is, for the model

$$g(a', b'; x_F) = a' x_F + b'.$$

Exercise 2.9

Implement a Python function `compute_optimal_params(x, y)` that computes and returns the optimal slope and intercept for a simple linear regression model using the closed-form expressions in Theorem 2.1.2. Specifically, your function should:

(a) Validate that the inputs `x` and `y` are one-dimensional `NumPy` arrays of equal length. If not, raise an error.

(b) Calculate and return the optimal slope and the optimal intercept using the formulas in Theorem 2.1.2.

(c) Calculate and print the predicted values $f(a^*, b^*; x_i)$ for each training input.

(d) Calculate and print the value of the empirical risk for the trained model.

(e) Generate and show a scatter plot of the training data along with the regression line.

(f) In your notebook, include two examples from synthetic datasets.

Your solution should be packaged as a self-contained `Jupyter` notebook. Refer to Example 2.3.3 and Example 2.4.2 for inspiration.

Exercise 2.10

Answer the following conceptual questions based on your understanding of the material in Section 2.2.

(a) How can we geometrically interpret the empirical risk in the context of the vector-valued constant model?

(b) Intuitively speaking why does it make sense for the mean of the training data to minimize the squared loss?

(c) *True or False*: The gradient of the empirical risk function is independent of the parameter vector $\vec{w}$.

Exercise 2.11

Suppose $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ is a dataset of vectors in $\mathbb{R}^2$. Let $\vec{\bar{x}}$ denote their mean, which minimizes the MSE of the constant model $f(\vec{w}; \vec{x}) = \vec{w}$. Let

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}.$$

Suppose we transform the data using $\mathbf{A}$, writing

$$\vec{x}_i' = \mathbf{A}\vec{x}_i.$$

(a) Find a formula for the new minimizer of the MSE for the constant model associated with the dataset $\{\vec{x}_1', \vec{x}_2', \ldots, \vec{x}_n'\}$.

(b) Let α denote the minimal value of the MSE associated with the original data. Find a formula for α', the minimal MSE of the constant model trained from the dataset $\{\vec{x}_1', \vec{x}_2', \ldots, \vec{x}_n'\}$, in terms of α.

Exercise 2.12

Suppose $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ is a dataset of vectors in $\mathbb{R}^2$. For the vector-valued constant model $f(\vec{w}; \vec{x}) = \vec{w}$ with parameter $\vec{w} \in \mathbb{R}^2$, define the absolute loss

$$L_{\text{abs}}(\vec{w}, \vec{x}) = |\vec{w}^{(1)} - \vec{x}^{(1)}| + |\vec{w}^{(2)} - \vec{x}^{(2)}|. \tag{19}$$

(a) Write the empirical risk of the absolute loss associated with the training data $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ as a sum of two functions R_1 and R_2, such that R_1 depends only on the *first* coordinates of the training data and parameter vector, and such that R_2 depends only on the second.

(b) Use Theorem 1.2.3 to prove that the minimizer of the empirical risk of the absolute loss is given by

$$\vec{w}^* = \begin{bmatrix} \text{median}\{\vec{x}_1^{(1)}, \vec{x}_2^{(1)}, \ldots, \vec{x}_n^{(1)}\} \\ \text{median}\{\vec{x}_1^{(2)}, \vec{x}_2^{(2)}, \ldots, \vec{x}_n^{(2)}\} \end{bmatrix}. \tag{20}$$

(c) Finally, using parts (a)-(b), explain how the loss function in Eq. (19) and the formula in Eq. (20) can be generalized to training data drawn from $\mathbb{R}^d$.

Exercise 2.13

Consider a multiple linear regression model trained on data $\{(\vec{x}_i, y_i)\}_{i=1}^n$ with features $\vec{x}_i \in \mathbb{R}^d$. Decide whether the following statements are true or false. Provide a brief explanation for your answer.

(a) After scaling all features $\vec{x}_i$ by a nonzero constant, both the optimal weight vector and the intercept will be scaled by the same constant.

(b) In multiple linear regression, the target variable is always a scalar.

(c) The intercept b is the predicted target when all feature components are zero.

(d) If the optimal parameter vector $\vec{w}^*$ satisfies $|(\vec{w}^*)^{(1)}| > |(\vec{w}^*)^{(2)}|$, then the model $f(\vec{w}^*, b^*; \vec{x})$ will be more sensitive to changes in the first feature coordinate than the second.

(e) The model $f(\vec{w}, b; \vec{x})$ contains d parameters.

(f) The mean-squared error of $f(\vec{w}^*, b^*; \vec{x})$ with respect to the optimal parameters $\vec{w}^*, b^*$ is equal to zero.

(g) The minimum mean-squared error of $f(\vec{w}^*, b^*; \vec{x})$ can only decrease with the addition of new training examples.

(h) The columns of the design matrix $\mathbf{Z}$ contain the training feature vectors $\vec{x}_i$.

Exercise 2.14

A chemist is analyzing how the yield of a chemical reaction (in grams) depends on three variables: concentration of the reactant (in mol/L), reaction time (in minutes), and catalyst amount (in mg). Her dataset is given below:

Concentration (mol/L)	$\vec{x}_i^{(1)}$	0.8	1.0	0.9	1.2	0.7	1.1	0.85	1.05
Reaction Time (min)	$\vec{x}_i^{(2)}$	30	35	32	40	28	37	31	36
Catalyst Amt. (mg)	$\vec{x}_i^{(3)}$	5	6	5.5	7	4.5	6.5	5.2	6.1
Yield (g)	y_i	50	65	60	80	45	70	55	68

(a) Calculate the parameters $\vec{w}^* \in \mathbb{R}^3, b^* \in \mathbb{R}$ which minimize the MSE for the multiple linear regression model $f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b$.

(b) Find the MSE associated to the optimal parameters and the chemist's dataset.

(c) Would you recommend using this model to the chemist? Why or why not?

Exercise 2.15

A materials scientist is studying how the strength of a composite material (in MPa) depends on three variables: fiber volume fraction (in percentage), curing time (in hours), and temperature during curing (in °C). His dataset is:

Fiber Volume (%)	$\vec{x}_i^{(1)}$	45	50	48	55	47	52	49	53
Curing Time (h)	$\vec{x}_i^{(2)}$	3	4	3.5	5	3	4.5	3.8	4.2
Temperature (°C)	$\vec{x}_i^{(3)}$	120	115	118	110	122	117	119	113
Strength (MPa)	y_i	150	160	155	170	148	165	152	168

(a) Calculate the parameters $\vec{w}^* \in \mathbb{R}^3, b^* \in \mathbb{R}$ which minimize the MSE for the multiple linear regression model $f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b$.

(b) Find the MSE associated to the optimal parameters and the scientist's dataset.

(c) Would you recommend using this model to the scientist? Why or why not?

Exercise 2.16

Let $\mathbf{A} \in \mathbb{R}^{p \times q}$ be any given matrix.

(a) Prove that $\ker(\mathbf{A}) = \ker(\mathbf{A}^\top \mathbf{A})$ as follows.

(i) If $x \in \ker(\mathbf{A})$, then $x \in \ker(\mathbf{A}^\top \mathbf{A})$.

(ii) If $x \in \ker(\mathbf{A}^\top \mathbf{A})$, then $x \in \ker(\mathbf{A})$.

(b) Use the rank-nullity theorem (see Theorem B.4) and (a) to show that $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}(\mathbf{A}^\top \mathbf{A})$.

(c) Consider a multiple linear regression model trained on data $\{(\vec{x}_i, y_i)\}_{i=1}^n$. Using (a) and (b), show that if the design matrix $\mathbf{Z} \in \mathbb{R}^{n \times (d+1)}$ has full column rank, then the normal equations $\mathbf{Z}^\top \mathbf{Z}\, \theta = \mathbf{Z}^\top y$ have a unique solution for the parameters θ.

Exercise 2.17

Raymondo and Gena are data scientists working on a multiple linear regression model to predict housing prices (in thousands USD) based on three features:

(i) size (ft^2),

(ii) age of the house (years),

(iii) number of bathrooms.

Their training data contain $n = 50$ observations. Suppose they *standardize* each

feature vector $\vec{x}_i \in \mathbb{R}^3$ to

$$\widetilde{\vec{x}}_i = \frac{\vec{x}_i - \overline{\vec{x}}}{\sigma}, \text{ where}$$
$$\overline{\vec{x}} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i,$$
$$\sigma = \left(\frac{1}{n}\sum_{i=1}^{n} \|\vec{x}_i - \overline{\vec{x}}\|^2\right)^{1/2}.$$

Raymondo claims the transformation will not change the model's accuracy after refitting, because it is invertible and can be "undone." Gena argues that refitting on the standardized features *will* change the predictions and make them more reliable.

(a) Let $\vec{w}^*, b^*$ be the optimal parameters obtained *before* standardization, and let $\widehat{\vec{w}}^*, \widehat{b}^*$ be those obtained *after* standardization. Derive closed-form expressions relating $(\widehat{\vec{w}}^*, \widehat{b}^*)$ to $(\vec{w}^*, b)$, $\overline{\vec{x}}$ and σ. (Hint: use the normal equation $\mathbf{Z}^\top \mathbf{Z}\vec{\theta}^* = \mathbf{Z}^\top \vec{y}$ and the fact that $\widetilde{\mathbf{Z}} = \sigma^{-1}\mathbf{Z} - \sigma^{-1}\mathbf{1}\overline{\vec{x}}^\top$.)

(b) Using (a) prove that the optimal MSE is invariant under standardization and refitting.

(c) After standardization, each coefficient $\vec{w}^{(j)}$ measures the expected change in the target variable for a 1-*standard-deviation* increase in the feature. Explain how this can help compare the relative importance of heterogeneous features such as 'size' and 'age.'

(d) Based on your explanation in parts (a) and (b), decide whether Raymondo or Gena is correct. Provide a detailed justification for your answer.

Exercise 2.18

Let $\{(\vec{x}_i, y_i)\}_{i=1}^n$ be a dataset with features $\vec{x}_i \in \mathbb{R}^d$. Suppose we wish to use the **intercept-free multiple linear regression model**:

$$f(\vec{w}; \vec{x}) = \vec{w}^\top \vec{x}.$$

(a) Write an expression for the empirical risk $R(\vec{w})$ associated to the square loss of the model and corresponding dataset.

(b) Compute the gradient $\nabla R(\vec{w})$ with respect to $\vec{w}$ by differentiating $R(\vec{w})$ with respect to each component of $\vec{w}$. Follow steps we used earlier in Example 2.3.1 and the proof of Theorem 2.3.2, with careful attention to how this scenario differs from before.

(c) Set the gradient equal to the zero vector and derive the resulting normal equations in matrix form. (Hint: Let $\mathbf{U} \in \mathbb{R}^{n\times d}$ be the matrix whose ith row is $\vec{x}_i^\top$ and $\vec{y} \in \mathbb{R}^n$ be the vector of targets – this resembles the design matrix but has a key difference.)

(d) Assuming that $\mathbf{U}^\top \mathbf{U}$ is invertible, solve the normal equations to obtain the closed-form solution for $\vec{w}^*$.

Exercise 2.19

Consider a multiple linear regression model with two features and no intercept:

$$f(\vec{w}; \vec{x}) = \vec{w}^\top \vec{x} = \vec{w}^{(1)} \vec{x}^{(1)} + \vec{w}^{(2)} \vec{x}^{(2)},$$

where the loss is defined using the absolute loss function

$$L(\vec{w}; (\vec{x}, y)) = \left| y - (\vec{w}^{(1)} \vec{x}^{(1)} + \vec{w}^{(2)} \vec{x}^{(2)}) \right|.$$

Assume that the training dataset is specially structured so that each observation has only one nonzero feature. That is, the dataset has the format

$$\left\{ \left(\begin{bmatrix} \vec{x}_1^{(1)} \\ 0 \end{bmatrix}, y_1 \right), \left(\begin{bmatrix} \vec{x}_2^{(1)} \\ 0 \end{bmatrix}, y_2 \right), \ldots, \left(\begin{bmatrix} \vec{x}_{n_1}^{(1)} \\ 0 \end{bmatrix}, y_{n_1} \right), \right.$$
$$\left. \left(\begin{bmatrix} 0 \\ \vec{x}_{n_1+1}^{(2)} \end{bmatrix}, y_{n_1+1} \right), \left(\begin{bmatrix} 0 \\ \vec{x}_{n_1+2}^{(2)} \end{bmatrix}, y_{n_1+2} \right), \ldots, \left(\begin{bmatrix} 0 \\ \vec{x}_{n_1+n_2}^{(2)} \end{bmatrix}, y_{n_1+n_2} \right) \right\}$$

with total observations $n = n_1 + n_2$.

(a) Show that under these assumptions the empirical risk can be written in the form

$$R(\vec{w}) = \frac{1}{n} \left[\sum_{i=1}^{n_1} \left| y_i - \vec{w}^{(1)} \vec{x}_i^{(1)} \right| + \sum_{i=n_1+1}^{n_1+n_2} \left| y_i - \vec{w}^{(2)} \vec{x}_i^{(2)} \right| \right].$$

(b) Use Theorem 1.2.3 to prove that the minimizer for each coordinate is given by

$$\begin{bmatrix} (\vec{w}^*)^{(1)} \\ (\vec{w}^*)^{(2)} \end{bmatrix} = \begin{bmatrix} \text{median} \left\{ \frac{y_i}{\vec{x}_i^{(1)}} : i = 1, \ldots, n_1 \right\} \\ \text{median} \left\{ \frac{y_i}{\vec{x}_i^{(2)}} : i = n_1 + 1, \ldots, n_1 + n_2 \right\} \end{bmatrix}$$

(c) Discuss why in the general case (when each observation has nonzero entries in both features) a closed-form solution for the optimal parameters using absolute loss is difficult to obtain, and explain how the solution (if it exists) might differ from the square loss solution.

Exercise 2.20

Implement a Python function `plot_mlr_results(x, y)` that solves the multiple linear regression problem for its inputs. Your function should:

(a) Validate that the input arrays `x` and `y` are compatible shapes. If not, raise an error.

(b) Calculate and return the optimal parameters using the formulas in Theorem 2.3.2. If $\mathbf{Z}^\top \mathbf{Z}$ is not invertible, raise an error.

(c) Print a table listing each input vector, the actual target, and the predicted target.

(d) Generate and show a scatter plot of actual versus predicted targets with proper labels and a legend.

(e) In your notebook, include two examples from synthetic datasets.

Your solution should be packaged as a self-contained `Jupyter` notebook. Refer to Example 2.3.3 and Example 2.4.2 for inspiration.

Exercise 2.21

A senior financial analyst at a major investment firm is developing a predictive model to estimate two key characteristics of stocks: the expected annual return (in %) and the annualized volatility (in %). To capture various financial aspects of the companies, the analyst selects four features, measured over the previous calendar year:

- **P/E Ratio**: The price-to-earnings ratio.
- **Dividend Yield (%)**: The annual dividend expressed as a percentage of the stock price.
- **Debt-to-Equity Ratio**: A measure of the company's financial leverage.
- **Beta**: The stock's volatility relative to the market.

She proposes to use a general linear model of the form

$$f(\mathbf{W}, \vec{b}; \vec{x}) = \mathbf{W}\vec{x} + \vec{b},$$

where $\vec{x} \in \mathbb{R}^4$, and with parameters $\mathbf{W} \in \mathbb{R}^{2\times 4}$ and $\vec{b} \in \mathbb{R}^2$. The target vector is comprised of the expected annual return and volatility. The analyst collects data for 8 different stocks, summarized in the table below.

P/E Ratio	$\vec{x}_i^{(1)}$	15	18	20	16	22	19	17	21
Dividend Yield (%)	$\vec{x}_i^{(2)}$	2.0	2.5	3.0	2.2	2.8	2.6	2.3	3.0
Debt-to-Equity	$\vec{x}_i^{(3)}$	0.50	0.70	0.40	0.60	0.80	0.55	0.65	0.75
Beta	$\vec{x}_i^{(4)}$	1.10	0.90	1.20	1.00	1.30	1.15	0.95	1.25
Expected Return (%)	$\vec{y}_i^{(1)}$	8.0	9.0	10.0	8.5	11.0	9.5	8.8	10.2
Volatility (%)	$\vec{y}_i^{(2)}$	5.0	5.5	6.0	5.2	6.3	5.8	5.1	6.0

(a) Find the design matrix $\mathbf{Z} \in \mathbb{R}^{8\times 5}$ and the target matrix $\mathbf{Y} \in \mathbb{R}^{8\times 2}$ from the data provided.

(b) Use Theorem 2.4.1 and a short `Python` calculation to find the weight matrix $\mathbf{W}^*$ and bias vector $\vec{b}^*$ which minimise MSE for the general linear model.

(c) Calculate the predicted target vectors for each stock and compute the overall MSE between the actual targets and the predictions.

(d) Use `Matplotlib` to draw a scatter plot of the predicted vs. actual targets.

(e) The analyst collects data from two new stocks, and reports her findings below.

P/E Ratio	$\vec{x}_i^{(1)}$	17	21
Dividend Yield (%)	$\vec{x}_i^{(2)}$	2.1	2.4
Debt-to-Equity	$\vec{x}_i^{(3)}$	0.49	0.73
Beta	$\vec{x}_i^{(4)}$	1.14	0.88
Expected Return (%)	$\vec{y}_i^{(1)}$	8.1	8.5
Volatility (%)	$\vec{y}_i^{(2)}$	8.5	7.1

Use your previously trained model, find the predicted values for expected return and volatility for the new stocks. How well does the model hold up? Decide if you will recommend the model to the analyst.

Exercise 2.22

(a) Use Theorem 2.4.1 with $d = p = 1$ to write a short proof of Theorem 2.1.2.

(b) Use Theorem 2.4.1 with $p = 1$ to write a short proof of Theorem 2.3.2.

Exercise 2.23

Let $\vec{x}, \vec{y} \in \mathbb{R}^n$ be non-zero vectors. Recall Theorem B.2(3) that

$$\|\vec{x}\|^2 = \sqrt{\vec{x}^\top \vec{x}} \geq 0.$$

Define

$$\vec{z} = \frac{\vec{x}}{\|\vec{x}\|} - \frac{\vec{y}}{\|\vec{y}\|}.$$

(a) Expand $\vec{z}^\top \vec{z}$ to obtain an expression involving $\vec{x}^\top \vec{y}$.

(b) Rearrange your expression to deduce the Cauchy-Schwarz inequality.

(c) Identify precisely when equality holds in the inequality above and interpret this geometrically. Make sure to review Theorem B.2.

Exercise 2.24

Consider the exponential decay model

$$f(a, b; x) = ae^{-bx},$$

with scalar-valued parameters a, b and scalar input-output pairs (x, y). We use the square loss function:

$$L_{\text{sq}}(a, b; (x, y)) = \left(y - ae^{-bx}\right)^2.$$

(a) Clearly state the empirical risk $R(a, b)$ associated with the dataset $\{(x_i, y_i)\}_{i=1}^n$.

(b) Compute the partial derivatives of $R(a, b)$ with respect to the parameters a and b, and simplify your expressions.

(c) Show that the normal equations are given by

$$\sum_{i=1}^n e^{-bx_i}\left(y_i - ae^{-bx_i}\right) = 0, \quad \text{and} \quad \sum_{i=1}^n x_i ae^{-bx_i}\left(y_i - ae^{-bx_i}\right) = 0.$$

(d) Isolate the parameter a explicitly in terms of b and the training data. You should obtain a closed-form expression for a.

(e) Substitute your expression for a from part (d) back into the equation involving b, and simplify as much as possible. Then clearly state that the optimal parameters a^*, b^* are given by:

$$a^* = \frac{\sum_{i=1}^n y_i e^{-b^* x_i}}{\sum_{i=1}^n e^{-2b^* x_i}},$$

where b^* satisfies the implicit equation:

$$\sum_{i=1}^n x_i e^{-b^* x_i}\left(y_i - a^* e^{-b^* x_i}\right) = 0.$$

Can you find b^* explicitly? What are some ways you could find b^* numerically?

(f) Show that the determinant of the Hessian matrix of $R(a, b)$ is given by

$$\det(H) = \frac{4}{n^2}\left[\left(\sum_{i=1}^n e^{-2bx_i}\right)\left(\sum_{i=1}^n a^2 x_i^2 e^{-2bx_i}\right) - \left(\sum_{i=1}^n a x_i e^{-2bx_i}\right)^2\right].$$

Then, using the Cauchy-Schwarz inequality, show that $\det(H)$ will be positive as long as the values $\{x_i\}_{i=1}^n$ are not all identical.

3 More on Modeling

> What theory of markets—other than barefoot empiricism—specifies such a model? Higher powers...make no substantive sense... What then can the regression coefficient on t^5 mean: "A $100,000,000$ quintic-year change in artist age explains a β_5 change in $\log_e$ prices?!"
>
> - *Edward Tufte*

So far we have covered the general framework of the modeling method in some detail, with particular emphasis on the linear model as a useful proving ground for the method and its related concepts. This chapter will hone in on a few different ways in which we can push these tools and theoretical concepts even further. Section 3.1 will explore basic nonlinear extensions of the linear models we have already seen, and show how the MLR pipeline can be extended to these settings. Section 3.2 will introduce convexity as a powerful tool for understanding the existence of minimizers, as well as the natural follow-up of gradient descent, which is a powerful tool for finding them. In Section 3.3 we will use creative risk functions to improve our catalog of modeling tools and, ideally, help reduce their failure points. At the end of this chapter, the reader will be ready to take on a variety of data science challenges.

3.1 Polynomial Regression and Interactions

In Chapter 2 we learned that for a training dataset of scalar-valued input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ the simple linear regression model

$$\begin{cases} f(a^*, b^*; x) & = a^* x + b^* \\ a^* & = \frac{\sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^n (x_i - \overline{x})^2}, \\ b^* & = \overline{y} - a^* \overline{x} \end{cases}$$

is the unique minimizer of the mean squared error provided the x_i's are not all identical. There are many examples we explored in which this model is relatively performant: predicting vehicle fuel economy as a function of horsepower, predicting sales revenue as a function of advertising investment, and so on. In many real-world situations, however, the relationship between input and output variables is significantly more nuanced. Fig. 2 illustrates a simple scenario where this is apparent: even data with a slightly nonlinear underlying trend can lead to complete failure of a simple linear model.

Underfitting refers to the situation where a model is too rigid to capture the salient structure in the data. This can often be seen when the error values for training and test data remain comparatively high when the model is situated alongside models with more trainable parameters. There are many conceivable ways to address this shortcoming of the linear model and perhaps the most obvious is to incorporate more diversity into our toolkit of models to include nonlinear ones as well; this has already been explored to

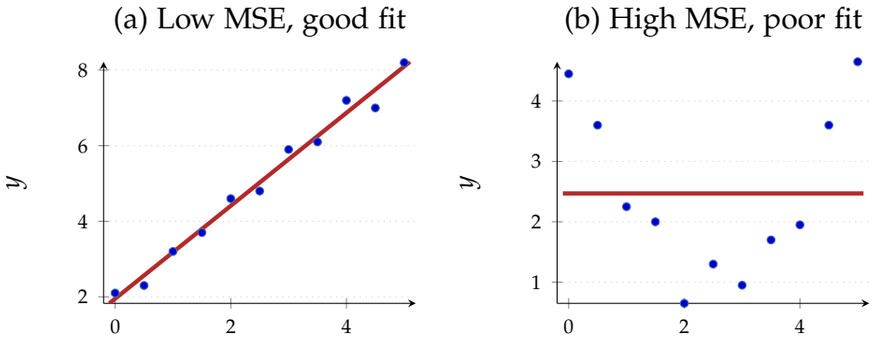


Figure 2: A side-by-side comparison of datasets for which the simple linear model is (a) appropriate and (b) clearly inappropriate.

some extent in Chapter 1 as well as the exercises at the end of Chapter 2. The good news is that in this setting, we can retool much of the pipeline of multiple linear regression to work for highly nonlinear models without a drastic investment in overhead.

The entry point to these methods is **polynomial regression**, which refers to designing and training predictive models whose formulas are given by polynomials in the input features. In essence, the idea is that where a *linear* model might fail to capture a reliable picture of the underlying data, by allowing quadratic, cubic, or even quartic terms, one can then start to build more robust and reliable models. This is illustrated in Fig. 3.

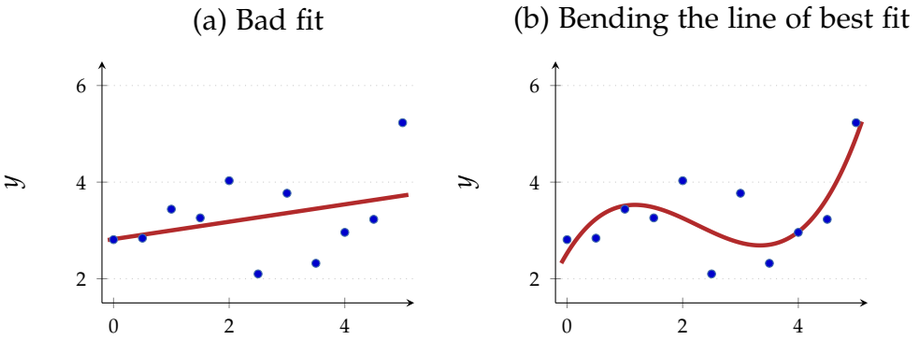


Figure 3: An illustration of "bending the curve" in order to achieve a more performant model.

To this end, the **simple polynomial regression model** with **degree** $d \geq 1$ is given by the model formula

$$f(\vec{\theta}; x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)}x + \vec{\theta}^{(2)}x^2 + \ldots + \vec{\theta}^{(d)}x^d. \tag{21}$$

Note: in this book we have "reserved" the letter d for the feature dimension of the data. In this setting we are being slightly abusive in also using it for the degree of the polynomial model, however, we hope this causes minimal confusion since this formula should look somewhat familiar: if we define the **polynomial feature vector** of x by the

formula

$$[\vec{x}] = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{bmatrix} \in \mathbb{R}^{d+1},$$

then Eq. (21) takes the form

$$f(\vec{\theta}; x) = \vec{\theta}^\top [\vec{x}].$$

Therefore, using the familiar mean-squared error:

$$R(\vec{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \vec{\theta}^\top [\vec{x}])^2, \tag{22}$$

all of our hard work from Chapter 2 pays off and we can essentially recast the task of minimizing $R(\vec{\theta})$ as instead minimizing mean squared error of a multiple linear regression model *in the features* $[\vec{x_i}]$. For this purpose, for a dataset of scalar-valued input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ we define the **transformed design matrix** by the formula

$$[\mathbf{Z}] = \begin{bmatrix} [\vec{x}_1]^\top \\ [\vec{x}_2]^\top \\ \vdots \\ [\vec{x}_n]^\top \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^d \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}.$$

With this setup, we have the following result.

Theorem 3.1.1 *Optimal Model Parameters for Simple Polynomial Regression*

Suppose we have a collection of training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i, y_i \in \mathbb{R}$ and model y using a simple polynomial regression model $f(\vec{\theta}; x)$ of the form

$$f(\vec{\theta}; x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} x + \vec{\theta}^{(2)} x^2 + \ldots + \vec{\theta}^{(d)} x^d = \vec{\theta}^\top [\vec{x}].$$

Let $[\mathbf{Z}]$ denote the polynomial design matrix associated to the training data and let $\vec{y} \in \mathbb{R}^n$ denote the vector of labels. Assume that $([\mathbf{Z}]^\top [\mathbf{Z}])^{-1}$ exists. Then the parameters $\vec{\theta}^* \in \mathbb{R}^{d+1}$ which minimize the mean squared error (see Eq. (22)) associated to the training data are unique and given by the formula

$$\vec{\theta}^* = ([\mathbf{Z}]^\top [\mathbf{Z}])^{-1} [\mathbf{Z}]^\top \vec{y}.$$

So far so good, right? However, there is a catch– our setup assumes and uses the invertibility of $[\mathbf{Z}]^\top [\mathbf{Z}]$ freely and although we have some intuition for when this is a feasible ask in the linear setting, it is not immediately clear when, if at all, we can expect this matrix to behave well in the nonlinear setting.

Theorem 3.1.2

Suppose we have a collection of training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i, y_i \in \mathbb{R}$. Then the polynomial design matrix has the property that $([\mathbf{Z}]^\top[\mathbf{Z}])^{-1}$ exists if and only if $n \geq d+1$ and there are at least $d+1$ distinct input training features.

Proof

This proof consists of two steps. First, assuming that $([\mathbf{Z}]^\top[\mathbf{Z}])^{-1}$ exists, by Exercise 2.16, it must hold that $\text{rank}([\mathbf{Z}]^\top[\mathbf{Z}]) = \text{rank}([\mathbf{Z}]) = d+1$ and therefore that $n \geq d+1$, since the rank of a matrix must never exceed either the number of its rows or columns. Similarly, if the set $\{x_i\}$ contained less than or equal to d distinct entries, then among its n rows there would be a sufficient amount of repetition to force $\text{rank}(([\mathbf{Z}])) \leq d$. But this would imply that $\text{rank}(([\mathbf{Z}])) \leq d$ is not full rank and that $[\mathbf{Z}]^\top[\mathbf{Z}]$ is noninvertible, a contradiction.

The second step starts by assuming on the other hand that $n \geq d+1$ and there are at least $d+1$ distinct input training features. Assume for contradiction that $[\mathbf{Z}]^\top[\mathbf{Z}]$ is noninvertible. Then, once again by Exercise 2.16, there must exist a nonzero vector $\vec{c} \in \mathbb{R}^{(d+1)}, \vec{c} \neq \vec{0}$ such that $[\mathbf{Z}]\vec{c} = \vec{0}$. Define the polynomial

$$p(t) = \vec{c}^{(0)} + \vec{c}^{(1)}t + \vec{c}^{(2)}t^2 + \ldots + \vec{c}^{(d)}t^d.$$

The condition $[\mathbf{Z}]\vec{c} = \vec{0}$, when expanded, asserts that

$$\sum_{j=1}^{d+1} [\mathbf{Z}]^{(i,j)}\vec{c}^{(j)} = 0, \qquad \text{for each } i = 1, \ldots, n.$$

But based on the construction of the polynomial design matrix, this means

$$\sum_{j=1}^{d+1} x_i^{(j)}\vec{c}^{(j)} = 0 \iff p(x_i) = 0,$$

i.e., that the polynomial $p(t)$ has a root at each point $t = x_i$. By assumption, there are at least $d+1$ distinct entries x_i, and thus $p(t)$ has at least $d+1$ zeros. The only polynomial of degree d with at least $d+1$ zeros is the constant zero polynomial. This implies that

$$\vec{c}^{(0)} = \vec{c}^{(1)} = \ldots = \vec{c}^{(d)} = 0,$$

a contradiction. Therefore $([\mathbf{Z}]^\top[\mathbf{Z}])^{-1}$ must exist.

It is perhaps worth mentioning that the matrix $[\mathbf{Z}]$ is an example of a *Vandermonde matrix*, for which many interesting mathematical problems have been studied.

In many applications, particularly when there is even a small amount of noise in the feature space, distinct x_i is typically a weak assumption. In this case it is natural to wonder: if a closed-form solution exists to the simple polynomial regression problem for any $d \leq n-1$, why not simply choose the highest degree possible, and in turn the most

flexible model we could possibly use? Such a modeling choice would greatly run the risk of **overfitting**, which occurs when a model is so flexible that it begins to memorize the noise peculiar to the training set. Moreover, small perturbations in the sample, or even random experimental noise, can lead to large swings in the learned parameters of the resulting model. Training error can plunge toward zero, but the test error, where applicable, typically rises. We can demonstrate this a numerical example below.

🍊 Example 3.1.3

In this synthetic example, consider the function

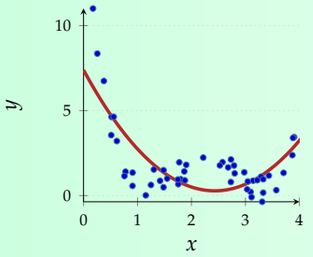
$$y = 16 - 35x + 28x^2 - 9x^3 + x^4.$$

Suppose we sample fifty points $\{(x_i, y_i)\}_{i=1}^{50}$ as follows: pick x_i uniformly at random from the interval $[0, 4]$, and then pick $y_i = 16 - 35x_i + 28x_i^2 - 9x_i^3 + x_i^4 + \epsilon_i$ where ϵ_i is a number sampled uniformly at random from $[-1, 1]$.

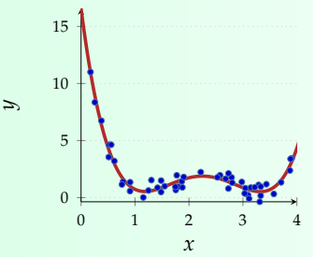
i	x_i	y_i
1	3.095	0.220
2	1.755	0.668
⋮	⋮	⋮
50	0.559	4.634

We can then fit several simple polynomial regression models to our data; in the illustrations below we show a scatterplot of the dataset alongside the corresponding curves of best fit for $d = 2, 4, 8$:

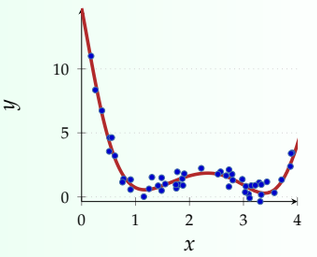
(a) $d = 2$, $MSE \approx 1.77$



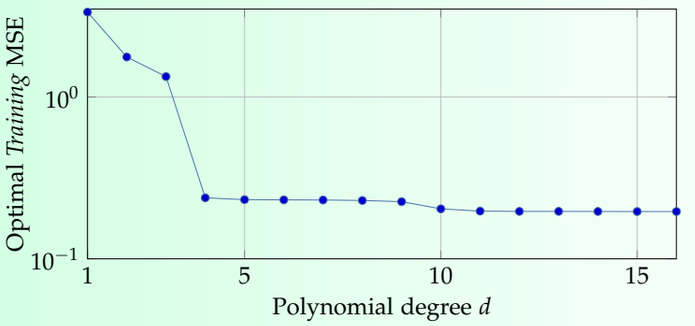
(b) $d = 4$, $MSE \approx .238$



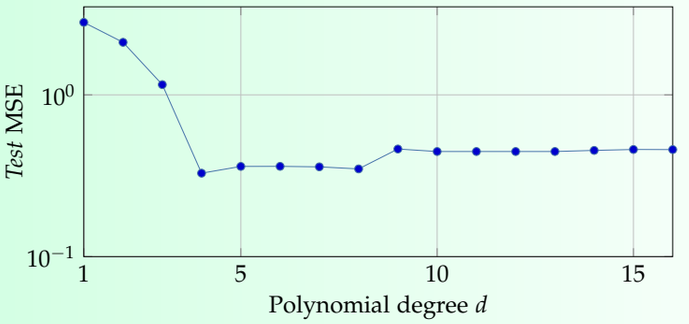
(c) $d = 8$, $MSE \approx .229$



We observe that the optimal mean-squared error is decreasing monotonically as we increase the model complexity. In fact, this trend more or less holds:



Despite a diminishing return as d gets above five or six, in theory, we could continue to see improvements until the optimal MSE reaches zero for $d = 49$ at which point the curve of best fit would perfectly interpolate each of the points. All of this sounds great, but here's where the tide turns: if we sample 25 *new* pairs $\{(x_i^{\text{test}}, y_i^{\text{test}})\}_{i=1}^{25}$ from the same data model as before and compute the MSE against the pre-trained models, we find that performance on these data is significantly worse:



This demonstrates the impact of overfitting, as well as the role of train-test data splits in mitigating the risk. By comparing these plots, $d = 4$ emerges as a likely candidate for a useful model; and indeed, as we saw at the beginning, this is correct.

We mention as well that Eq. (21) and Theorem 3.1.1 extend naturally to settings where polynomial terms x^t are replaced by linearly independent functions; for example, one could consider the model formula

$$f(\vec{\theta}; x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} e^{-x} + \vec{\theta}^{(2)} e^{-2x} + \ldots + \vec{\theta}^{(d)} e^{-dx}, \tag{23}$$

or any suitably rich mixture of functions that are relevant to a given modeling scenario. This essentially only leads to a change in the setup of the design matrix, a modification which we will explore in the exercises. Simple polynomial regression, as well as the more exotic example in Eq. (23) are instances of models that are linear in the data after applying a **data transformation**.

Finally we consider how this setup affects the multivariate setting by returning to multiple linear regression. Suppose we have a training dataset of vector-valued input features and scalar-valued targets $\{(\vec{x}_i, y_i)\}_{i=1}^n$. When the response depends not only on individual features but also on their joint effects, we can extend the standard multiple linear regression model to include what are known as **interaction terms**, which are weighted products of various features. For example, a multiple linear regression model with $d = 2$ and a single interaction term might look like

$$f(\vec{w}, v, b; \vec{x}) = \vec{w}^{(1)} \vec{x}^{(1)} + \vec{w}^{(2)} \vec{x}^{(2)} + v \vec{x}^{(1)} \vec{x}^{(2)} + b, \qquad \vec{w} \in \mathbb{R}^2,\ v, b \in \mathbb{R}.$$

In this case $f(\vec{w}, v, b; \vec{x})$ contains four parameters and scales quadratically on the input data. Some examples where interaction terms might be useful in expanding a model's robustness include: modeling drug outcomes (e.g., concentration of bacteria) as a function which is not only linear in the dosage amount of Drug A and Drug B, but also their product; or modeling the amount of plant growth in a biology experiment as a function of the amount of water and fertilizer a plant receives as well as their product. Note, however, that introducing unnecessary interaction terms without proper domain knowledge or theoretical justification can lead to the same overfitting problems as before.

To minimize the MSE of a multiple linear regression model with respect to a given dataset, we can apply the same procedures we have seen in this section so far.

Theorem 3.1.4 *Optimal Model Parameters for MLR with Interaction*

Suppose we have a collection of training data $\{(\vec{x}_i, y_i)\}_{i=1}^n$ where $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Let $I = \{(i_1, j_1), \ldots, (i_m, j_m)\}$ be a collection of $m \geq 1$ pairs of indices with $i_t \neq j_t$ for $t = 1, \ldots, m$. Suppose we model y using a multiple linear regression model $f(\vec{w}, \vec{v}, b; \vec{x})$ containing interaction terms from I of the form

$$f(\vec{w}, \vec{v}, b; \vec{x}) = \vec{w}^\top \vec{x} + \sum_{t=1}^{m} \vec{v}^{(i_t, j_t)} \vec{x}^{(i_t)} \vec{x}^{(j_t)} + b.$$

Let $[\![\mathbf{Z}]\!]$ denote the **interaction design matrix** given by

$$[\![\mathbf{Z}]\!] = \begin{bmatrix} 1 & \vec{x}_1^{(1)} & \cdots & \vec{x}_1^{(d)} & \vec{x}_1^{(i_1)} \vec{x}_1^{(j_1)} & \cdots & \vec{x}_1^{(i_m)} \vec{x}_1^{(j_m)} \\ 1 & \vec{x}_2^{(1)} & \cdots & \vec{x}_2^{(d)} & \vec{x}_2^{(i_1)} \vec{x}_2^{(j_1)} & \cdots & \vec{x}_2^{(i_m)} \vec{x}_2^{(j_m)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & \vec{x}_n^{(1)} & \cdots & \vec{x}_n^{(d)} & \vec{x}_n^{(i_1)} \vec{x}_n^{(j_1)} & \cdots & \vec{x}_n^{(i_m)} \vec{x}_n^{(j_m)} \end{bmatrix} \in \mathbb{R}^{n \times (1+d+m)}$$

and let $\vec{y} \in \mathbb{R}^n$ denote the vector of labels. Assume that $([\![\mathbf{Z}]\!]^\top [\![\mathbf{Z}]\!])^{-1}$ exists. Then the parameters $\vec{w}^* \in \mathbb{R}^d$, $\vec{v}^* \in \mathbb{R}^m$, and $b^* \in \mathbb{R}$ which minimize the mean squared error associated to the training data are unique and given by the formula

$$\begin{bmatrix} b^* \\ \vec{w}^* \\ \vec{v}^* \end{bmatrix} = ([\![\mathbf{Z}]\!]^\top [\![\mathbf{Z}]\!])^{-1} [\![\mathbf{Z}]\!]^\top \vec{y}.$$

Example 3.1.5

An NBA analyst argues that a player's scoring output (measured in points per game, denoted PPG) depends not only on the features minutes per game (MPG), usage rate (USG, which measures the percentage of team plays a player uses while on the court), and three-point attempts per game (3PA); but *also* on an interaction term between MPG and USG. The analyst collects the following data for ten players, averaged over several games in the current season:

Player	MPG	USG	3PA	MPG×USG	PPG
i	$\vec{x}_i^{(1)}$	$\vec{x}_i^{(2)}$	$\vec{x}_i^{(3)}$	$\vec{x}_i^{(1)}\vec{x}_i^{(2)}$	y_i
1	34	30	7	1020	27
2	28	22	4	616	18
3	15	18	1	270	6
4	25	27	6	675	20
5	32	25	3	800	24
6	18	14	2	252	8
7	30	21	5	630	19
8	12	20	1	240	5
9	20	17	3	340	10
10	35	29	8	1015	29

With $d = 3$ features, one interaction term, and an intercept, the analyst fits the model

$$f(\vec{w}, v, b; \vec{x}) = \vec{w}^\top \vec{x} + v\vec{x}_i^{(1)}\vec{x}_i^{(2)} + b.$$

The interaction design matrix and label vector are given by

$$[\![\mathbf{Z}]\!] = \begin{bmatrix} 1 & 34 & 30 & 7 & 1020 \\ 1 & 28 & 22 & 4 & 616 \\ 1 & 15 & 18 & 1 & 270 \\ 1 & 25 & 27 & 6 & 675 \\ 1 & 32 & 25 & 3 & 800 \\ 1 & 18 & 14 & 2 & 252 \\ 1 & 30 & 21 & 5 & 630 \\ 1 & 12 & 20 & 1 & 240 \\ 1 & 20 & 17 & 3 & 340 \\ 1 & 35 & 29 & 8 & 1015 \end{bmatrix}, \qquad \vec{y} = \begin{bmatrix} 27 \\ 18 \\ 6 \\ 20 \\ 24 \\ 8 \\ 19 \\ 5 \\ 10 \\ 29 \end{bmatrix}.$$

After some `Python` computations guided by Theorem 3.1.4, the final model is trained and given by

$$f(\vec{w}, v, b; \vec{x}) \approx 0.594\vec{x}^{(1)} + 0.304\vec{x}^{(2)} + 0.154\vec{x}^{(3)} + 0.007\vec{x}_i^{(1)}\vec{x}_i^{(2)} - 9.97.$$

3.2 Convexity & Gradient Descent

Our modeling journey thus far has exposed us to a variety of model formulas, loss and risk functions, and applications thereof. However a key feature of the material presented up to this point has been the existence of closed-form analytical solutions to the corresponding risk minimization problems. In practice, this is often a lot to ask for: in most real use cases, the optimal solutions to the risk minimization problems are both impossible to represent using pen-and-paper linear algebra-based formulas as well as elusive to find using brute force numerical methods (e.g., grid searches). With that in mind, the focus of this section is two powerful tools which allow us to guarantee the existence of risk function minimizers and find them in practice.

Our first topic is convexity. We begin with some definitions. First, recall that a collection of points $C \subseteq \mathbb{R}^n$ is said to be a **convex set** provided that for each $\vec{x}, \vec{y} \in C$, the straight line segment between them is also contained in C; in other words, we have

$$\lambda\vec{x} + (1-\lambda)\vec{y} \in C, \text{for each } \lambda \in [0,1].$$

This definition should have a familiar tone, particularly in the context of grade school geometry and the notions of convex and non-convex polygons (see Fig. 4).

Consider next a function $f : \mathbb{R}^n \to \mathbb{R}$. We say that f is a **convex function** if its **epigraph** (the set of points on and above its graph) $\{(\vec{\theta}, t) \mid f(\vec{\theta}) \leq t\} \subseteq \mathbb{R}^{n+1}$ is a convex set. Equivalently, f is a convex function if for each $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^n$ and $\lambda \in [0,1]$ it holds that

$$f\big((1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2\big) \;\leq\; (1-\lambda)f(\vec{\theta}_1) + \lambda f(\vec{\theta}_2).$$

Figure 4: Examples of a convex set (left) and a non-convex set (right) in $\mathbb{R}^2$.

We say that f is a **strictly convex function** if the inequality above is strict except at the endpoints:

$$f\big((1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2\big) \;<\; (1-\lambda)f(\vec{\theta}_1) + \lambda f(\vec{\theta}_2), \quad \lambda \in (0,1).$$

These notions are illustrated in Fig. 5.

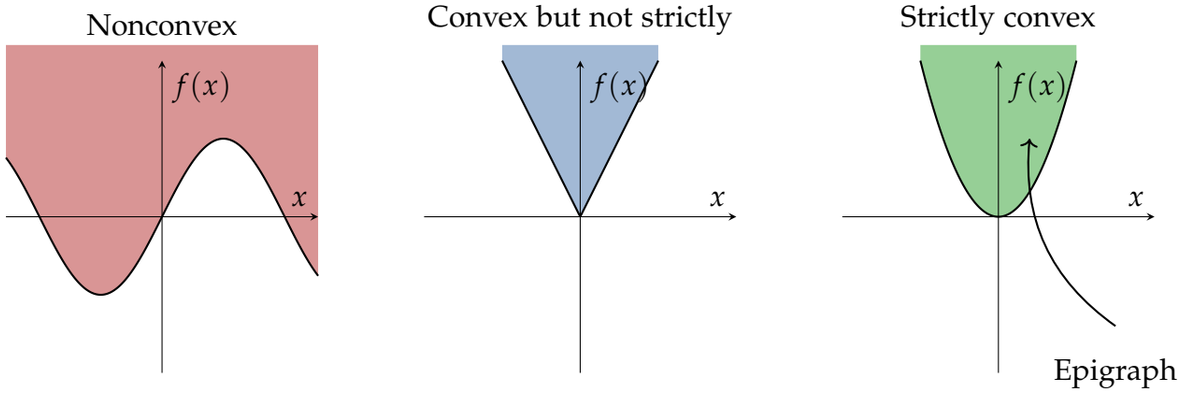


Figure 5: Examples of nonconvex (left, $f(x) = \sin(x)$), convex but not strictly convex (center, $f(x) = |x|$), and strictly convex (right, $f(x) = x^2$) functions on $\mathbb{R}$.

As one can start to imagine based on the illustrations provided, convex functions have strong properties when it comes to minimizers.

Theorem 3.2.1 *Minimizers of convex functions*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then:

(i) If $\vec{\theta}^* \in \mathbb{R}^n$ is a local minimizer of f, it must also be a global minimizer of f.

(ii) The set of global minimizers

$$\mathcal{M} = \Big\{\vec{\theta} \in \mathbb{R}^n : \vec{\theta} \text{ is a minimizer of } f\Big\}$$

is a convex subset of $\mathbb{R}^n$.

(iii) If f is also strictly convex, then there is at most one global minimizer (i.e., any minimizer is unique if it exists).

(iv) Lastly, if f is also differentiable on $\mathbb{R}^n$ and $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^n$, it holds that

$$f(\vec{\theta}_2) - f(\vec{\theta}_1) \;\geq\; \nabla f(\vec{\theta}_1)^\top (\vec{\theta}_2 - \vec{\theta}_1).$$

Moreover,

$$\nabla f(\vec{\theta}^*) = \vec{0} \iff \vec{\theta}^* \in \mathcal{M}.$$

Proof

For statement (i), suppose for the purposes of contradiction that $\vec{\theta}^* \in \mathbb{R}^n$ is a local minimizer but not global. Then there exists $\vec{y} \in \mathbb{R}^n$ such that

$$f(\vec{y}) < f(\vec{\theta}^*).$$

Consider the line segment $\{(1-\lambda)\vec{\theta}^* + \lambda\vec{y} : \lambda \in [0,1]\}$. Since f is assumed minimal in a neighborhood of $\vec{\theta}^*$, i.e. at the point on the line segment for $\lambda = 0$, it must hold that for sufficiently small $\lambda > 0$,

$$f((1-\lambda)\vec{\theta}^* + \lambda\vec{y}) > f(\vec{\theta}^*).$$

On the other hand, convexity also gives

$$\begin{aligned} f((1-\lambda)\vec{\theta}^* + \lambda\vec{y}) &\leq (1-\lambda)f(\vec{\theta}^*) + \lambda f(\vec{y}) \\ &< (1-\lambda)f(\vec{\theta}^*) + \lambda f(\vec{\theta}^*) = f(\vec{\theta}^*). \end{aligned}$$

which is a contradiction. For statement (ii), let $\vec{\theta}_1, \vec{\theta}_2$ be two global minimizers, so $f(\vec{\theta}_1) = f(\vec{\theta}_2) = m = \min f$. Then for any $\lambda \in [0,1]$, by convexity of f, it holds that

$$f\big((1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2\big) \;\leq\; (1-\lambda)f(\vec{\theta}_1) + \lambda f(\vec{\theta}_2) = m.$$

But m is the global minimum, so equality holds and $(1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2$ is also a minimizer. Therefore the minimizer set is convex. For statement (iii), if f is strictly convex and had two distinct minimizers $\vec{\theta}_1 \neq \vec{\theta}_2$, then for any $\lambda \in (0,1)$

$$m = f\big((1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2\big) < (1-\lambda)f(\vec{\theta}_1) + \lambda f(\vec{\theta}_2) = m,$$

contradicting minimality. Hence at most one minimizer exists. Finally, for statement (iv), let $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^n$ be fixed, and consider the univariate function

$$\phi(\lambda) = f\big((1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2\big), \quad \lambda \in [0,1].$$

Since f is convex on $\mathbb{R}^n$, it follows that $\lambda \mapsto (1-\lambda)\vec{\theta}_1 + \lambda\vec{\theta}_2$ is affine, ϕ is also convex and differentiable on $[0,1]$. In particular, for any $t \in [0,1]$ it holds that

$$\phi((1-t)(0) + t\lambda) \leq (1-t)\phi(0) + t\phi(\lambda)$$

This implies that

$$\phi'(0) = \lim_{t \to 0^+} \frac{\phi(t\lambda) - \phi(0)}{t} \leq \phi(\lambda) - \phi(0)$$

Next, noting that

$$\phi(1) = f(\vec{\theta}_2), \quad \phi(0) = f(\vec{\theta}_1), \quad \text{and} \quad \phi'(0) = \nabla f(\vec{\theta}_1)^\top(\vec{\theta}_2 - \vec{\theta}_1),$$

it holds that

$$f(\vec{\theta}_2) - f(\vec{\theta}_1) \;\geq\; \nabla f(\vec{\theta}_1)^\top (\vec{\theta}_2 - \vec{\theta}_1).$$

Therefore, if $\vec{\theta}^* \in \mathbb{R}^n$ satisfies $\nabla f(\vec{\theta}^*) = \vec{0}$ and $\vec{\theta} \in \mathbb{R}^n$ is any other point, we have

$$f(\vec{\theta}) - f(\vec{\theta}^*) \;\geq\; \vec{0}^\top (\vec{\theta}^* - \vec{\theta}) = 0 \iff f(\vec{\theta}^*) \leq f(\vec{\theta})$$

which implies $\vec{\theta}^* \in \mathcal{M}$.

These properties illustrate why convexity is so useful for data scientists. In particular, there are essentially no distinctions between local minimizers and global minimizers once they have been found; so, for example, if an algorithm is designed only to find local minima, in this case it is also discovering global minima, in which we are most often interested.

To *verify* that a function is convex, there are many avenues to take. One example is the second derivative test, which we will explore further in the exercises. We will conclude this portion of the section by showing that two of our most used risk functions are in fact convex, and in some cases strictly so.

🍊 Example 3.2.2

Suppose we have a training dataset of vector-valued input features and scalar-valued targets $\{(\vec{x}_i, y_i)\}_{i=1}^n$ (we assume the targets are scalar-valued only for simplicity; this example extends to the vector-valued case easily). Suppose that $f(\vec{\theta}; \vec{x})$ is a given model , where $\vec{\theta} \in \mathbb{R}^m$ is a parameter vector of arbitrary length. We assume f is *affine* in $\vec{\theta}$ (such as a simple linear model, multiple regression model, or any instance where f is linear in θ after possibly applying a transformation to x). Recall the mean-squared error

$$R_{sq}(\vec{\theta}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - f(\vec{\theta}; \vec{x})|^2.$$

Then $R_{sq}(\vec{\theta})$ is convex. To see this, for $i = 1, \ldots, n$ define the residual

$$r_i(\vec{\theta}) = y_i - f(\vec{\theta}; \vec{x}_i), \qquad R_{sq}(\vec{\theta}) = \sum_{i=1}^{n} r_i(\vec{\theta})^2.$$

Since $f(\vec{\theta}; \vec{x}_i)$ is affine in $\vec{\theta}$, for any two parameter vectors $\vec{\theta}_1, \vec{\theta}_2$ and any $\lambda \in [0, 1]$ we have

$$r_i\big(\lambda\vec{\theta}_1 + (1-\lambda)\vec{\theta}_2\big) = \lambda\, r_i(\vec{\theta}_1) + (1-\lambda)\, r_i(\vec{\theta}_2).$$

Hence since $x \mapsto x^2$ is convex on $\mathbb{R}$,

$$\begin{aligned}\big[r_i(\lambda\vec{\theta}_1 + (1-\lambda)\vec{\theta}_2)\big]^2 &= \big[\lambda\, r_i(\vec{\theta}_1) + (1-\lambda)\, r_i(\vec{\theta}_2)\big]^2 \\ &\leq \lambda\, r_i(\vec{\theta}_1)^2 + (1-\lambda)\, r_i(\vec{\theta}_2)^2.\end{aligned}$$

Summing over $i = 1, \ldots, n$ and dividing by n gives

$$R_{sq}\big(\lambda\vec{\theta}_1 + (1-\lambda)\vec{\theta}_2\big) \;\leq\; \lambda\, R_{sq}(\vec{\theta}_1) + (1-\lambda)\, R_{sq}(\vec{\theta}_2),$$

which is exactly the definition of convexity for R_{sq}. Therefore R_{sq} is convex in $\vec{\theta}$. The exact same argument works for showing that the mean absolute error

$$R_{abs}(\vec{\theta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\vec{\theta};\, \vec{x})|$$

is also convex; and, for that matter, so is the empirical risk associated to any p loss.

So far, we have set the stage by explaining that a certain class of functions make for good risk functions since their minimizers are all around great *once they have been found*. But how do we find them? For the remainder of this section we will focus on such a technique, which is called the **gradient descent algorithm**. This is an iterative method which allows one to track down minimizers based on an initial guess and information about the behavior of the function nearby. Before defining the algorithm itself, let's start by reviewing a result from multivariable calculus which is essential for understanding why this method works so well.

Theorem 3.2.3

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function and let $\vec{\theta}_0 \in \mathbb{R}^n$ be a fixed vector. Then the direction along which f increases most rapidly is $\nabla f(\vec{\theta}_0)$ and the direction along which f decreases most rapidly is $-\nabla f(\vec{\theta}_0)$.

Proof

Let $\vec{\theta}_0 \in \mathbb{R}^n$ be given and suppose $\vec{u} \in \mathbb{R}^n$ is any unit vector, $\|\vec{u}\| = 1$. The directional derivative of f at $\vec{\theta}_0$ in the direction $\vec{u}$ is given by

$$D_{\vec{u}} f(\vec{\theta}_0) = \lim_{t \to 0^+} \frac{f(\vec{\theta}_0 + t\,\vec{u}) - f(\vec{\theta}_0)}{t} = (\nabla f(\vec{\theta}_0))^\top \vec{u}.$$

By the Cauchy-Schwarz inequality,

$$|(\nabla f(\vec{\theta}_0))^\top \vec{u}| \leq \|\nabla f(\vec{\theta}_0)\|\, \|\vec{u}\| = \|\nabla f(\vec{\theta}_0)\|.$$

Equality holds precisely when $\vec{u}$ points in the same direction as $\nabla f(\vec{\theta}_0)$, i.e. $\vec{u} = \frac{\nabla f(\vec{\theta}_0)}{\|\nabla f(\vec{\theta}_0)\|}$. Thus f increases most rapidly in the direction $\nabla f(\vec{\theta}_0)$. Similarly, by replacing $\vec{u}$ with $-\vec{u}$ one sees that the minimum directional derivative is attained when $\vec{u} = -\frac{\nabla f(\vec{\theta}_0)}{\|\nabla f(\vec{\theta}_0)\|}$.

The basic idea here is that if you are trying to minimize a function f and you are situated at a candidate location $\vec{\theta}_0$, Theorem 3.2.3 suggests that you should "move your

guess in the direction $-\nabla f(\vec{\theta}_0)$" in order to find a new candidate which will exhibit a lower value of f. An illustration of this concept is given in Fig. 6.

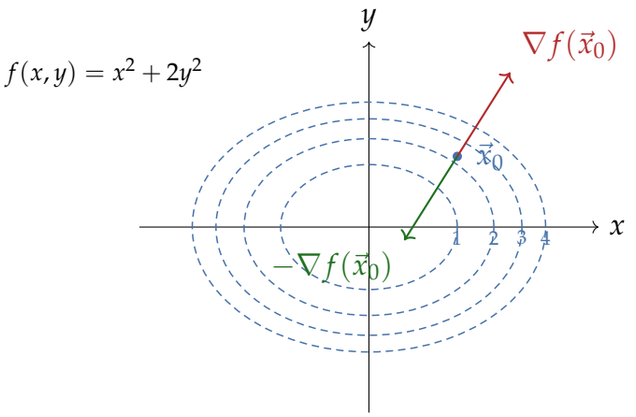


Figure 6: An illustration of Theorem 3.2.3 using the function $f(x,y) = x^2 + 2y^2$. Contour lines at $z = 1,2,3,4$ are shown dashed blue lines, along with a sample point $\vec{x}_0 = (1, 0.8)^\top$ and $\pm\nabla f(\vec{x}_0)$.

With this in mind, we can fully state the gradient descent algorithm.

Gradient Descent

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a given differentiable function, $\eta > 0$ a fixed positive scalar known as the **learning rate**, $\vec{\theta}_0 \in \mathbb{R}^n$ a fixed initial point, and let $T \geq 0$ be a fixed number of steps.

At step $t \geq 1$, set $\vec{\theta}_t$ according to the update rule

$$\vec{\theta}_t = \vec{\theta_{t-1}} - \eta(\nabla f(\vec{\theta_{t-1}})). \tag{24}$$

Return $\vec{\theta}_T$.

There are many ways to analyze the convergence and long-term behavior of this algorithm; for our purposes, we will state but not prove one of the more basic convergence results which appears in the gradient descent literature.

Theorem 3.2.4 *Convergence of gradient descent*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex and differentiable function whose gradient satisfies the inequality

$$\|\nabla f(\vec{\theta}_1) - \nabla f(\vec{\theta}_2)\| \leq L\,\|\vec{\theta}_1 - \vec{\theta}_2\|, \qquad \forall\, \vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^n.$$

Fix an initial point $\vec{\theta}_0 \in \mathbb{R}^n$ and a learning rate η satisfying $0 < \eta < \frac{2}{L}$. Generate the sequence $\{f(\vec{\theta}_t)\}_{t=0}^{\infty}$ according to Eq. (24). Then the sequence $\{f(\vec{\theta}_t)\}_{t=0}^{\infty}$ converges to a local minimizer of f.

The intuition that Theorem 3.2.4 lends itself to is that for many "suitably well-behaved" functions f, as long as the learning rate $\eta > 0$ is *not too large*, the gradient descent algorithm will converge. Picking η to be inappropriately large often leads to sequences of points $\vec{\theta}_t$ which jump around in unpredictable ways. Moreover, as we saw in Theorem 3.2.1, if f is moreover assumed convex and we have found a local minimizer via Theorem 3.2.4, we know it must also be a global one.

Gradient descent also leads to a zoo of different modifications including adaptive learning rates, creative ways to calculate the gradient, and the usage of higher-order information about the behavior of f to inform directional choices. For now, let's conclude this section with some applications of gradient descent to simple calculus problems as well as a risk minimization problem we saw in the previous chapter.

Example 3.2.5 ***Gradient descent in action***

Consider the two-variable function $f : \mathbb{R}^2 \to \mathbb{R}$ given by the formula

$$f(\vec{\theta}) \;=\; e^{2(\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2}, \qquad \vec{\theta} = \begin{bmatrix} \vec{\theta}^{(1)} \\ \vec{\theta}^{(2)} \end{bmatrix} \in \mathbb{R}^2.$$

Note that f is convex and has a unique minimizer at the point $\vec{\theta}^* = \vec{0}$. Its gradient is given by

$$\nabla f(\vec{\theta}) \;=\; \begin{bmatrix} 4\,\vec{\theta}^{(1)} e^{2(\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2} \\ 2\,\vec{\theta}^{(2)} e^{2(\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2} \end{bmatrix}.$$

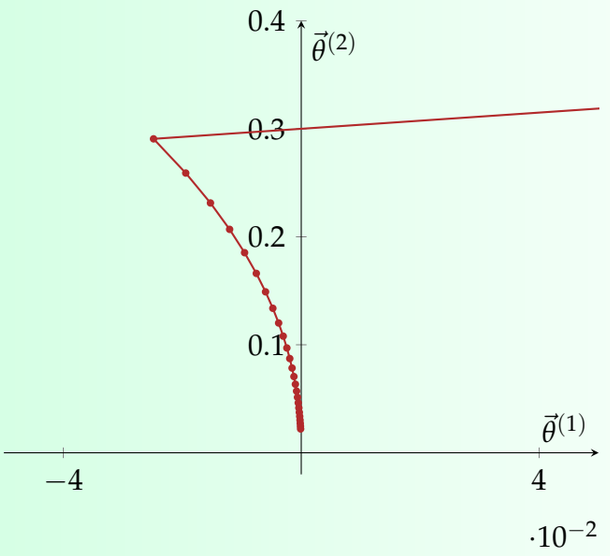
Starting from the nearby point $\vec{\theta}_0 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}$ and using a learning rate $\eta = 0.05$, the gradient-descent update

$$\vec{\theta}_{t+1} \;=\; \vec{\theta}_t - \eta\, \nabla f(\vec{\theta}_t)$$

produces the first five iterates shown below:

t	$\vec{\theta}_t^{\top}$	$\big(\nabla f(\vec{\theta}_t)\big)^{\top}$	$f(\vec{\theta}_t)$
0	$[\,0.800,\ 0.600\,]$	$[\,16.496,\ 6.186\,]$	5.155
1	$[\,-0.024,\ 0.290\,]$	$[\,-0.108,\ 0.633\,]$	1.089
2	$[\,-0.019,\ 0.259\,]$	$[\,-0.083,\ 0.554\,]$	1.070
3	$[\,-0.015,\ 0.231\,]$	$[\,-0.064,\ 0.488\,]$	1.055
4	$[\,-0.012,\ 0.206\,]$	$[\,-0.050,\ 0.432\,]$	1.044

Note that the values of f are decreasing throughout this process, and the points $\vec{\theta}_t$ are slowly converging towards $\vec{\theta}^* = \vec{0}$. We illustrate the path of the gradient descent algorithm in $\mathbb{R}^2$ below (initial point not shown):



Gradient descent is a powerful tool even when applied to situations where the objective is not quite perfect; take, for example, the empirical risk associated with absolute loss. This function is nondifferentiable at many candidate points and its derivative, when it exists, is not exactly the most straightforward to write down or evaluate. Are there workarounds to this issue? Yes, as we explain in the example below, and gradient descent tends to take them in stride.

Example 3.2.6 *Gradient descent and simple linear regression*

Suppose we start with six scalar input-output pairs

$$\{(x_i, y_i)\}_{i=1}^{6} = \{(0, 2.0),\ (1, 2.8),\ (2, 3.9),\ (3, 5.9),\ (4, 7.8),\ (5, 10.2)\}.$$

and we wish to use a simple linear model

$$f(\vec{\theta};\, x) = \vec{\theta}^{(1)} x + \vec{\theta}^{(2)}$$

with parameter vector $\vec{\theta} \in \mathbb{R}^2$. The mean absolute error is given by

$$\begin{aligned} R_{\text{abs}}(\vec{\theta}) &= \frac{1}{6} \sum_{i=1}^{6} \left| y_i - \left(\vec{\theta}^{(1)} x_i + \vec{\theta}^{(2)} \right) \right| \\ &= \tfrac{1}{6} \Big(|\, 2 - \vec{\theta}^{(2)}| + |\, 2.8 - (\vec{\theta}^{(1)} + \vec{\theta}^{(2)})| + \ldots + |10.2 - 5\vec{\theta}^{(1)} - \vec{\theta}^{(2)}| \Big) . \end{aligned}$$

Because the absolute value function is not differentiable at 0, we approximate each partial derivative of R_{abs} with a **finite difference approximation** as follows. Fix $t_0 > 0$ small (in this case $t_0 = 10^{-4}$) and use the approximation

$$\frac{\partial R_{\text{abs}}}{\partial \vec{\theta}^{(1)}}(\vec{\theta}_0) = \lim_{t \to 0} \frac{R_{\text{abs}}(\vec{\theta}_0 + t\vec{e}_1) - R_{\text{abs}}(\vec{\theta}_0)}{t} \approx \frac{R_{\text{abs}}(\vec{\theta}_0 + t_0\vec{e}_1) - R_{\text{abs}}(\vec{\theta}_0 - t_0\vec{e}_1)}{2t_0}.$$

where $\vec{e}_1$ is the first standard basis vector $\vec{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. In other words, we approximate the derivative using its limit definition and a choice of t which is very small.

This approximation always exists since even though the absolute function is non-differentiable, it is continuous, and therefore its difference quotient is always defined as long as the denominator is nonzero. Let $\widehat{\nabla R_{\text{abs}}}$ denote the finite-difference approximate gradient.

Next, we initialize the gradient descent loop at $\vec{\theta}_0 = [1 \;\; 1]^\top$ and set the learning rate to $\eta = 0.1$. For $t = 0, 1, \ldots, 5$ the first five iterates are

t	$\vec{\theta}_t^\top$	$\widehat{\nabla R_{\text{abs}}}(\vec{\theta}_t)^\top$	$R_{\text{abs}}(\vec{\theta}_t)$
0	$[1.000 \;\; 1.000]$	$[-2.500 \;\; -1.000]$	1.933
1	$[1.250 \;\; 1.100]$	$[-2.500 \;\; -1.000]$	1.208
2	$[1.500 \;\; 1.200]$	$[-1.833 \;\; -0.667]$	0.583
3	$[1.683 \;\; 1.267]$	$[0.833 \;\; 0.333]$	0.458
4	$[1.600 \;\; 1.233]$	$[-0.500 \;\; 0.000]$	0.433

After a few dozen iterations, the parameters stabilize near $\vec{\theta}^* \approx [1.92 \;\; 0.90]^\top$, which yields a mean absolute error below 0.05 on this toy dataset. We include some `Python` code below for an example implementation of this method.

```
# example data
x = np.array([0, 1, 2, 3, 4, 5], dtype=float)
y = np.array([2.0, 2.8, 3.9, 5.9, 7.8, 10.2], dtype=float)
n = len(x)

# hyperparameters
eta   = 0.1      # learning rate
h     = 1e-4     # finite-difference step
steps = 100      # number of GD iterations

# helper methods
def model(theta, x):
    return theta[0] * x + theta[1]

def mae(theta):
    return np.abs(y - model(theta, x)).mean()

def approx_grad(theta):
    grad = np.zeros_like(theta)
    for j in range(len(theta)):
        e = np.zeros_like(theta)
        e[j] = 1.0
        grad[j] = (mae(theta + h * e) - mae(theta - h * e)) / (2 * h)
    return grad

# gradient-descent loop
theta = np.array([1.0, 1.0])
history = [(0, *theta, mae(theta))]
```

```
for t in range(1, steps + 1):
    theta -= eta * approx_grad(theta)
    if t <= 5 or t == steps:
        history.append((t, *theta, mae(theta)))

# results
print("t   theta1   theta2   MAE")
for row in history:
    print(f"{row[0]:<3d} {row[1]:>7.3f}  {row[2]:>7.3f}  {row[3]:>6.3f}")
```

3.3 Regularization: Ridge and Lasso Regression

In Section 3.1 we saw how high-degree polynomials or families of interaction terms can drive the training error practically to zero, only to watch the test error rebound dramatically. This is essentially due to what we described as *overfitting*, in which the data scientist is essentially bending the model around their training data with excess precision leading to a model which captures not an underlying trend but simply the noise observed in the original training sample. This leads to unstable optimal weights, unreliable predictions, and frankly bad models. **Regularization** is one approach to resolving this instability and refers to augmenting the risk function with a **penalty** that discourages extremely large weights.

Two penalty families dominate modern practice. **Ridge regression** adds a squared vector length penalty which is governed by a regularization parameter $\mu > 0$; for example, a ridge risk function might look like

$$R(\vec{\theta}) + \mu \|\vec{\theta}\|^2,$$

where in this example $\vec{\theta}$ is a generic set of model parameters and R is any risk function. The idea here is that for the new and improved objective, an optimal set of weights $\vec{\theta}^*$ has not only low mean squared error but also generally small components; avoiding a scenario where $\vec{\theta}^*$ has perfect error but must take on extreme values in its entries in order to perfectly interpolate the training data. Note that in this context μ is acting as a **hyperparameter**, which is a configuration value set before training (e.g., a learning rate η, or number of iterations) that governs the behavior of the learning algorithm.

On the other hand, **Lasso regression** swaps the Euclidean vector length penalty for an absolute value penalty that might look like:

$$R(\vec{\theta}) + \mu \sum_{i=1}^{m} |\vec{\theta}^{(i)}|.$$

Risk functions such as this one often produce sparse models with zeroed weights that are easier to interpret. They have the downside, however, of usually being nondifferentiable and requiring more sophisticated techniques to optimize.

Caution!

When applying regularization terms to risk functions, we usually omit the intercept (where relevant) from the regularization term to avoid unnecessarily adjusting the intercept of our model.

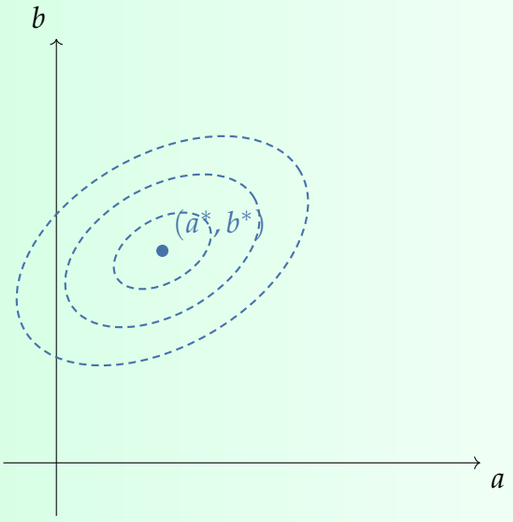
There are of course entire books written on these topics; the purpose of this section is to work through some theoretical results and numerical examples which allow us to gain some early intuition for these methods and their advantages. Our first step is an example which will hopefully shed some light on why the two penalty families behave differently.

Example 3.3.1 *Visualizing Regularization*

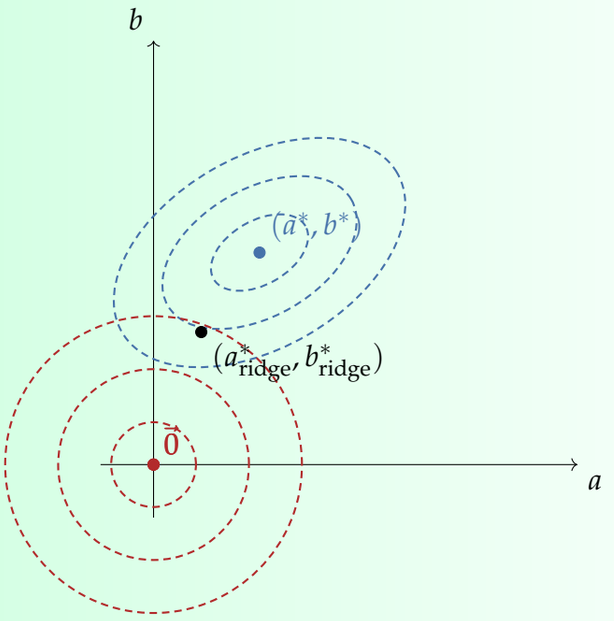
Consider the multiple linear regression model as trained on a collection of input-output pairs $\{(\vec{x}_i, y_i)\}_{i=1}$, where $\vec{x}_i \in \mathbb{R}^2$. Assume the intercept term is $b = 0$ for simplicity. The original model $f(a, b; \vec{x}) = a\vec{x}^{(1)} + b\vec{x}^{(2)}$ leads to a (mean-squared error) risk function

$$R(a, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (a\vec{x}^{(1)} + b\vec{x}^{(2)}))^2.$$

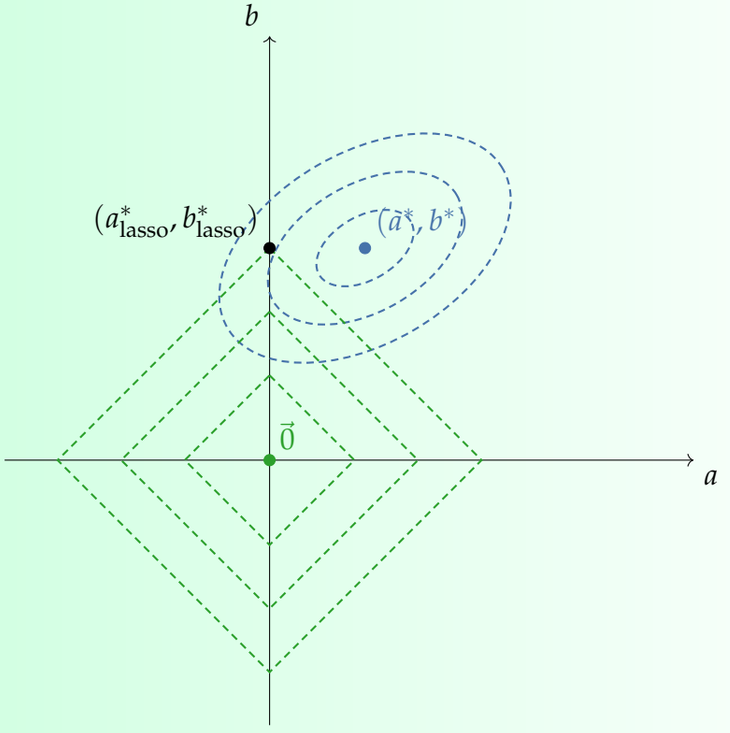
It is important to emphasize that when viewing the training data as fixed scalars, the risk function is simply a function of two variables a, b and can be visualized more or less in the same manner as other such functions. In particular, it admits a contour diagram.



When the only objective on hand is $R(a, b)$, the minimum in question is found by solving the normal equations and identifying a unique minimizer. When a regularization term enters the picture, things change. Take for example (with $\mu = 1$) the Regularization term $\| \begin{bmatrix} a \\ b \end{bmatrix} \|^2 = a^2 + b^2$. Note that this also admits a contour diagram, and has a unique minimizer at the origin; we can overlay it on top of the earlier one:



Thus, the minimizer $(a^*_{\text{ridge}}, b^*_{\text{ridge}})$ of the ridge regression problem can be thought of as trying to make both objective functions as small as possible, proportionally to the weight $\mu > 0$ which controls how far away from the original minimizer we wish the solution to be. On the other hand, if we instead consider the lasso penalty (again with $\mu = 1$) $|a| + |b|$, the contour lines of the penalty while, while still centered at the origin, change:



Instead of circles centered at the origin, the level curves take on the shape of diamonds. In three and higher dimensions, these become cubes and hypercubes.

A subtle observation is that the level curves of $R(a, b)$, as they spread out from the minimizers (a^*, b^*), will more quickly intersect the "points" or "corners" of the level curves of the penalty $|a| + |b|$ which lie on the axes–*precisely where one or more coordinates will equal zero.* It is for this reason that lasso penalties often (but not always) lead to optimal parameters which have one or more zeros.

Next we will revisit the multiple linear regression model from earlier and show that the solution to the ridge regression problem exists and is, in general, unique.

Theorem 3.3.2 ***Optimal parameters for ridge regression and MLR***

Let $\{(\vec{x}_i, y_i)\}_{i=1}^n$ be a collection of training data with $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Let $\mu > 0$ be a fixed hyperparameter, and suppose we model y using a multiple linear regression model $f(\vec{w}, b; \vec{x})$ of the form

$$f(\vec{w}, b; \vec{x}) = \vec{w}^\top \vec{x} + b.$$

Recall the design vectors $\vec{z}_i$ and the design matrix $\mathbf{Z}$ as usual. Let $\vec{\theta}^\top = \begin{bmatrix} b & \vec{w} \end{bmatrix}^\top$, and define the ridge risk

$$R_\mu(\vec{\theta}) = \frac{1}{n} \sum_{i=1}^n (y_i - \vec{\theta}^\top \vec{z}_i)^2 + \mu \|\vec{w}\|^2$$

Then, for each $\mu > 0$, the unique minimizer is given by

$$\vec{\theta}_\mu^* = \left(\mathbf{Z}^\top \mathbf{Z} + n\mu \widetilde{\mathbf{I}_{d+1}} \right)^{-1} \mathbf{Z}^\top \vec{y}.$$

Here, $\widetilde{\mathbf{I}_{d+1}}$ is the $(d+1) \times (d+1)$ identity matrix with the first entry set equal to zero.

Proof

Since $\vec{\theta} \mapsto \mu \|\vec{\theta}\|^2$ is strictly convex (this will be included as an exercise), $R_\mu(\vec{\theta})$ is a strictly convex function and therefore if $\nabla R_\mu(\vec{\theta}^*) = \vec{0}$ then $\vec{\theta}^*$ will be a unique global minimizer. Thus we can approach this problem as usual, by looking for a solution to the equation $\nabla R_\mu(\vec{\theta}^*) = \vec{0}$. We have already computed the gradient of the mean squared error in the previous chapter:

$$\nabla R_{sq}(\vec{\theta}) = -\frac{2}{n} \mathbf{Z}^\top (\vec{y} - \mathbf{Z}\vec{\theta}),$$

and it is relatively straightforward (by expanding the vector length) to see that

$$\nabla \mu \|\vec{w}\|^2 = 2\mu \vec{w},$$

therefore we have

$$\nabla R_\mu(\vec{\theta}) = -\frac{2}{n} \mathbf{Z}^\top (\vec{y} - \mathbf{Z}\vec{\theta}) + 2\mu \widetilde{\mathbf{I}_{d+1}} \vec{\theta}.$$

Note that $\widetilde{\mathbf{I}_{d+1}}\vec{\theta}$ is simply the vector $\vec{\theta}$ with the first entry removed, i.e., $\vec{w}$. Setting this gradient to $\vec{0}$ and rearranging yields the normal equations

$$\left(\tfrac{1}{n}\mathbf{Z}^\top\mathbf{Z} \;+\; \mu\,\widetilde{\mathbf{I}_{d+1}}\right)\vec{\theta} \;=\; \tfrac{1}{n}\mathbf{Z}^\top\vec{y} \iff \left(\mathbf{Z}^\top\mathbf{Z} + n\mu\,\widetilde{\mathbf{I}_{d+1}}\right)\vec{\theta} \;=\; \mathbf{Z}^\top\vec{y}.$$

We claim that the matrix $\mathbf{A} = \mathbf{Z}^\top\mathbf{Z} + n\mu\,\widetilde{\mathbf{I}_{d+1}}$ is invertible. Since it is square, it is enough to show that its kernel consists only of the zero vector. Suppose $\mathbf{A}\vec{v} = \vec{0}$ for some $\vec{v} \in \mathbb{R}^{d+1}$. Left-multiplying by $\vec{v}^\top$ and using symmetry gives

$$\begin{aligned} 0 \;=\; \vec{v}^\top\mathbf{A}\vec{v} &= \vec{v}^\top\mathbf{Z}^\top\mathbf{Z}\vec{v} \;+\; n\mu\,\vec{v}^\top\widetilde{\mathbf{I}_{d+1}}\vec{v} \\ &= \|\mathbf{Z}\vec{v}\|^2 \;+\; n\mu\|\widetilde{\mathbf{I}_{d+1}}\vec{v}\|^2. \end{aligned}$$

The rightmost term can only equal zero if $\widetilde{\mathbf{I}_{d+1}}\vec{v} = \vec{0}$, which means that all entries of $\vec{v}$ except possibly the first entry (which gets killed by $\widetilde{\mathbf{I}_{d+1}}$), must equal zero. If this is the case, based on the definition of $\mathbf{Z}$ (in particular, the leading one in each row), we would have

$$\vec{v} = \begin{bmatrix} \vec{v}^{(1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \implies \mathbf{Z}\vec{v} = \begin{bmatrix} 1 & \vec{x}_1^\top \\ 1 & \vec{x}_2^\top \\ \vdots & \\ 1 & \vec{x}_n^\top \end{bmatrix} \begin{bmatrix} \vec{v}^{(1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{v}^{(1)} \\ \vec{v}^{(1)} \\ \vdots \\ \vec{v}^{(1)} \end{bmatrix} \in \mathbb{R}^n.$$

But then the only way that $\|\mathbf{Z}\vec{v}\|^2 = 0$ as well is if each entry of the vector above is zero, i.e., if $\vec{v} = \vec{0}$ after all. Thus the kernel of $\mathbf{A}$ is trivial and it must therefore have an inverse. Solving for $\vec{\theta}$ therefore gives the unique minimizer

$$\vec{\theta}^*_\mu \;=\; \left(\mathbf{Z}^\top\mathbf{Z} + n\mu\,\widetilde{\mathbf{I}_{d+1}}\right)^{-1}\mathbf{Z}^\top\vec{y},$$

as claimed.

Note that this pipeline works just as well out-of-the-box when data transformations are used and we consider instead a multiple regression model in the transformed features. We demonstrate this pipeline in action in the following example.

Example 3.3.3

The analytics staff of an NHL franchise would like to forecast the number of goals a team will score in a game, y, using only the shot volume, x, it generates in a given game. Coaches expect that adding shots is beneficial up to a point, after which fatigue and lower-quality opportunities dampen the return, while very high volumes often come when the opponent is already worn down. The resulting "rise-plateau-rise" intuition suggests a weakly cubic pattern. By observing the performance of the franchise over several seasons and averaging the goal return for various numbers of shots, an analyst collected the following dataset of training observations:

x_i (shots)	10	15	20	25	30	35	40	45	50	55
y_i (average goals)	5.56	8.10	9.74	10.70	11.21	11.50	11.79	12.30	13.26	14.90

To explore the effect of ridge regression and help mitigate overfitting, the analysts consider the model

$$f(\vec{\theta};\, x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} x + \vec{\theta}^{(2)} x^2 + \vec{\theta}^{(3)} x^3$$

together with the risk function

$$R_\mu(\vec{\theta}) = \frac{1}{10} \sum_{i=1}^{10} (y_i - f(\vec{\theta};\, x_i))^2 + \mu((\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2 + (\vec{\theta}^{(3)})^2), \quad \mu > 0.$$

The corresponding design matrix $[\mathbf{Z}]$ is given by

$$[\mathbf{Z}] = \begin{bmatrix} 1 & 10 & 100 & 1000 \\ 1 & 15 & 225 & 3375 \\ 1 & 20 & 400 & 8000 \\ 1 & 25 & 625 & 15625 \\ 1 & 30 & 900 & 27000 \\ 1 & 35 & 1225 & 42875 \\ 1 & 40 & 1600 & 64000 \\ 1 & 45 & 2025 & 91125 \\ 1 & 50 & 2500 & 125000 \\ 1 & 55 & 3025 & 166375 \end{bmatrix},$$

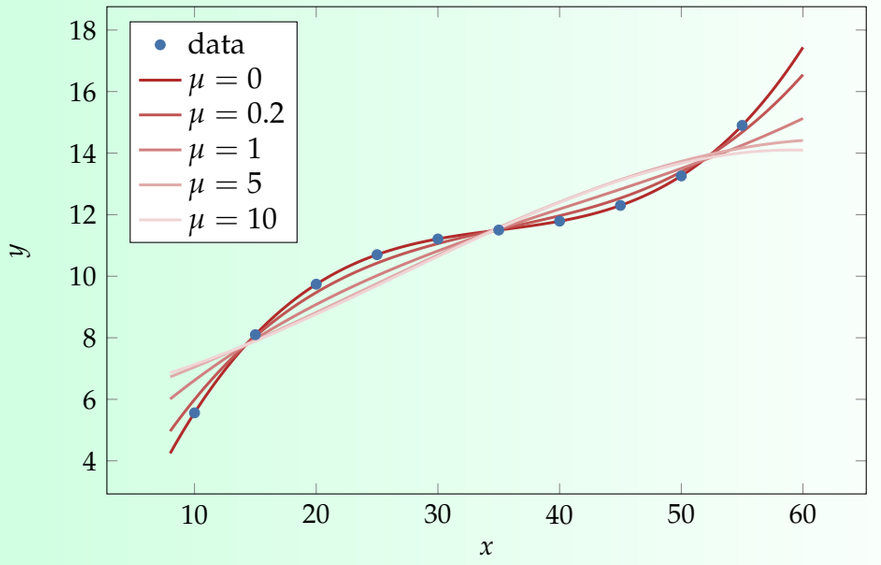
and thus by Theorem 3.3.2, the optimal parameter vector is given by is

$$\vec{\theta}^{\,*}_\mu \;=\; \left([\mathbf{Z}]^\top [\mathbf{Z}] + 10\mu\, \widetilde{\mathbf{I}_4}\right)^{-1} [\mathbf{Z}]^\top \vec{y},$$

giving, for various values of μ, the weights

μ	$\vec{\theta}^{(0)}$	$\vec{\theta}^{(1)}$	$\vec{\theta}^{(2)}$	$\vec{\theta}^{(3)}$
0	-3.112	1.152	-0.032	0.000
0.2	-0.579	0.852	-0.022	0.000
1	3.091	0.417	-0.007	0.000
5	5.620	0.118	0.003	-0.000
10	6.089	0.062	0.005	-0.000

We illustrate the curves of best fit in the plot below. The analyst observes that for μ small, the weights tend to be larger in general but also more heterogeneous, with some components significantly larger than others; for μ large, the coefficients become more homogeneous and lower in scale overall.



Example 3.3.4

The performance analytics team for a Grand Tour cycling squad wishes to predict the finishing time of a rider on a given stage, y_i (in hours), from three key features: the stage length $\vec{x}_i^{(1)}$ (in km), total elevation gain $\vec{x}_i^{(2)}$ (in meters), and average temperature $\vec{x}_i^{(3)}$ (in °C). Sparsity in the model parameters is desirable, since some features may have negligible impact on time. The team collects data from five recent stages:

stage i	$\vec{x}_i^{(1)}$ (km)	$\vec{x}_i^{(2)}$ (m gain)	$\vec{x}_i^{(3)}$ (°C)	y_i (hours)
1	150	2500	20	4.02
2	160	3000	18	4.30
3	170	2800	22	4.15
4	140	2200	23	3.85
5	155	2600	19	4.08

The team fits the multiple linear regression model

$$f(\vec{\theta};\, \vec{x}) \;=\; \vec{\theta}^{(0)} + \vec{\theta}^{(1)}\vec{x}_i^{(1)} + \vec{\theta}^{(2)}\vec{x}_i^{(2)} + \vec{\theta}^{(3)}\vec{x}_i^{(3)}, \qquad \vec{\theta} \in \mathbb{R}^4.$$

The design matrix is given by

$$\mathbf{Z} \;=\; \begin{bmatrix} 1 & 150 & 2500 & 20 \\ 1 & 160 & 3000 & 18 \\ 1 & 170 & 2800 & 22 \\ 1 & 140 & 2200 & 23 \\ 1 & 155 & 2600 & 19 \end{bmatrix}.$$

Since the designers wish to promote weight sparsity and prevent overfitting, they opt to use lasso regression and introduce the risk function

$$R_\mu(\vec{\theta}) \;=\; \frac{1}{5}\sum_{i=1}^{5}\left(y_i - \vec{\theta}^{\top}\vec{z}_i\right)^2 \;+\; \mu\sum_{j=1}^{3}\left|\vec{\theta}^{(j)}\right|,$$

Because R_μ is not differentiable in closed form, as in Example 3.2.6, we approximate its gradient by finite differences. For a small $t_0 > 0$ set

$$\frac{\partial R_\mu}{\partial \vec{\theta}^{(j)}} \approx \frac{R_\mu(\vec{\theta} + t_0\,\vec{e}_j) \;-\; R_\mu(\vec{\theta} - t_0\,\vec{e}_j)}{2t_0}, \quad j = 0, 1, 2, 3.$$

and let $\widehat{\nabla R_\mu}$ denote the approximate gradient of R_μ according to the componentwise definition above. Taking $t_0 = 10^{-4}$, step size $\eta = 10^{-7}$, and running $T = 2 \times 10^5$ iterations of finite difference gradient descent produces the following approximate minimizers $\vec{\theta}^*$ for various values of μ:

μ	$\vec{\theta}^{*(0)}$	$\vec{\theta}^{*(1)}$	$\vec{\theta}^{*(2)}$	$\vec{\theta}^{*(3)}$
0	2.961	−0.001	0.001	−0.007
0.1	2.660	0.000	0.001	0.000
1	2.692	0.000	0.001	0.000
5	2.834	0.000	0.000	0.000
10	3.012	0.000	0.000	0.000

After some further analysis, the team makes the following observations. First, when little or no lasso term is incorporated into the risk function, each of the three features contributes to the model: adjusting for scale, it seems that stage length and elevation gain are reasonably comparable, and that temperate seems to be somewhat more predictive of stage duration when compared to the others. However, when a stronger lasso term is incorporated into the risk, the temperature and stage length features fall away and elevation is revealed as the strongest surviving feature. This might make sense; if the stage lengths are all relatively close to their mean and temperature is fairly noisy, perhaps the most predictive feature when the model is constrained in its flexibility is in fact elevation gain. Lastly, as μ gets larger (in this case, approaching 10) we observe the tendency of the lasso penalty to drive coefficients to zero.

3.4 Bonus: Constrained optimization

Key Idea

This is a *bonus section* and contains some extra material which will not appear on the final exam.

Our modeling journey has treated most parameters as *free agents*: if minimizing the empirical risk asked us to set $\theta = -10^3$ we dutifully complied. Reality, of course, is rarely so accommodating. Many scenarios require the model parameters to conform to various geometric or mathematical patterns. Imagine you run a small logistics platform that must move digital photographs, printed books, and boxed merch from a handful of "source" data centers and warehouses to an equally small collection of "destination" retail nodes.

Every megabyte or mile has a cost, and—crucially—each location has a *hard budget* on what can leave or arrive. Ignoring those limits would produce an "optimal" plan that ships more cargo than a warehouse owns or delivers negative packages to a store: nonsense that a spreadsheet might gladly output if we forget to write down the constraints.

Back in Section 3.2 we learned that an unconstrained minimizer of a function $f(\vec{\theta})$ is found by hunting for a point where *all* partial derivatives vanish. When the feasible set is *restricted* by one or more equations

$$g_k(\vec{\theta}) \;=\; 0, \qquad k = 1, \dots, m,$$

the gradients of f and the g_k's must strike a delicate balance: at the optimum they become parallel. This geometric fact is codified by attaching a real-valued weight λ_k— a **Lagrange multiplier**—to each constraint and folding everything into one scalar function,

$$\mathcal{L}(\vec{\theta}, \lambda_1, \dots, \lambda_m) \;=\; f(\vec{x}) + \sum_{k=1}^{m} \lambda_k \, g_k(\vec{x}).$$

Finding stationary points of $\mathcal{L}$ gives you the candidate minimizers.

Theorem 3.4.1 *Lagrange Multiplier Theorem*

(1) Suppose $f, g : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable. Assume $\vec{\theta}^* \in \mathbb{R}^n$ satisfies the constraint

$$g(\vec{\theta}^*) \;=\; 0,$$

and that $\vec{\theta}^*$ is a (local) minimizer of f *subject to* $g(\vec{x}) = 0$. If

$$\nabla g(\vec{\theta}^*) \;\neq\; \vec{0},$$

then there exists a real number λ^*—the **Lagrange multiplier**—such that

$$\nabla f(\vec{\theta}^*) \;+\; \lambda^* \, \nabla g(\vec{\theta}^*) \;=\; \vec{0}.$$

Equivalently, the gradients ∇f and ∇g are parallel at the constrained optimum.

(2) If $g_1, \dots, g_m : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable and $\operatorname{rank}\big[\nabla g_1(\vec{\theta}^*) \;\cdots\; \nabla g_m(\vec{\theta}^*)\big] = m$, then there exist $\lambda_1^*, \dots, \lambda_m^* \in \mathbb{R}$ such that

$$\nabla f(\vec{\theta}^*) + \sum_{k=1}^{m} \lambda_k^* \, \nabla g_k(\vec{\theta}^*) = \vec{0}.$$

The proof of this theorem requires some more advanced multivariable calculus (namely the implicit function theorem) than is desirable for us to cover, so we omit a proof.[1]

Assuming we wish to minimize a differentiable function (such as a risk function) $f : \mathbb{R}^n \to \mathbb{R}$ subject to a single equality constraint

$$g(\vec{\theta}) \;=\; 0,$$

[1] You can find one here.

The Lagrange Multiplier Theorem converts this *constrained* problem into the familiar task of finding an *unconstrained* critical point. To do so, we introduce a new scalar variable $\lambda \in \mathbb{R}$ and define

$$\mathcal{L}(\vec{\theta}, \lambda) \;=\; f(\vec{\theta}) + \lambda\, g(\vec{\theta}).$$

Then we set all partial derivatives to zero:

$$\nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}, \lambda) = \vec{0}, \qquad \frac{\partial \mathcal{L}}{\partial \lambda} \;=\; g(\vec{\theta}) \;=\; 0.$$

These $n+1$ equations form a linear or nonlinear system for $(\vec{\theta}, \lambda)$. Each solution furnishes a *candidate* optimum. Convexity or a second-derivative test tells whether a candidate is the minimizer we want.

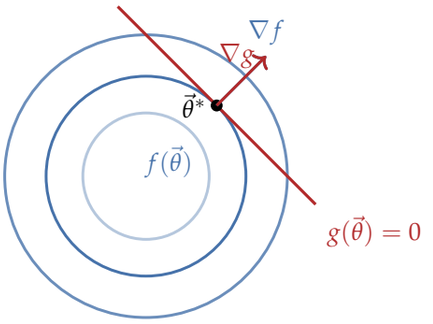


Figure 7: An illustration of the Lagrange multiplier theorem: on a feasible region (red) a minimizer $\vec{\theta}^*$ is found when the contour of f (blue) is tangent to the feasible region, i.e., the gradients of f, g are parallel.

Geometrically speaking, $\nabla f(\vec{\theta}^*)$ is perpendicular to the level-set of f, while $\nabla g(\vec{\theta}^*)$ is perpendicular to the constraint. Tangency therefore forces the two vectors to be parallel, exactly the relation guaranteed by the Lagrange multiplier equations.

Example 3.4.2 ***Point on a Line***

Suppose we wish to minimize the squared distance to the origin,

$$f(\vec{\theta}) \;=\; (\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2, \qquad \vec{\theta} = \begin{bmatrix} \vec{\theta}^{(1)} \\ \vec{\theta}^{(2)} \end{bmatrix} \in \mathbb{R}^2,$$

subject to the *linear* constraint

$$g(\vec{\theta}) \;=\; \vec{\theta}^{(1)} + \vec{\theta}^{(2)} - 1 \;=\; 0.$$

The Lagrangian is formed by writing

$$\mathcal{L}(\vec{\theta}, \lambda) \;=\; (\vec{\theta}^{(1)})^2 + (\vec{\theta}^{(2)})^2 + \lambda\big(\vec{\theta}^{(1)} + \vec{\theta}^{(2)} - 1\big).$$

This yields the equations

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \vec{\theta}^{(1)}} &= 2\,\vec{\theta}^{(1)} + \lambda = 0,\\ \frac{\partial \mathcal{L}}{\partial \vec{\theta}^{(2)}} &= 2\,\vec{\theta}^{(2)} + \lambda = 0,\\ \frac{\partial \mathcal{L}}{\partial \lambda} &= \vec{\theta}^{(1)} + \vec{\theta}^{(2)} - 1 = 0.\end{aligned}$$

From the first two equations $\vec{\theta}^{(1)} = \vec{\theta}^{(2)}$ and hence $2\,\vec{\theta}^{(1)} = 1 \implies \vec{\theta}^{(1)} = \vec{\theta}^{(2)} = \frac{1}{2}$. Substituting back gives $\lambda = -1$. The result is

$$\vec{\theta}^{*} \;=\; \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \qquad f(\vec{\theta}^{*}) \;=\; 0.5.$$

Example 3.4.3 *Estimating judges' weights in gymnastics scoring*

In many gymnastics competitions the final score y shown to the audience is a weighted average of three components reported by different judging panels:

Form	$\vec{x}^{(1)}$	execution quality (fewest deductions),
Difficulty	$\vec{x}^{(2)}$	declared difficulty value of the routine,
Style / Artistry	$\vec{x}^{(3)}$	overall aesthetic impression.

The meet regulations keep the exact weights secret, revealing only that they are non-negative and sum to 1. A coach has collected training data $\{(\vec{x}_i, y_i)\}_{i=1}^{n}$ where $\vec{x}_i = \big(\vec{x}_i^{(1)}, \vec{x}_i^{(2)}, \vec{x}_i^{(3)}\big)^{\top}$ are the three sub-scores for routine i (on a scale from zero to ten) and y_i is the published final score. She wishes to recover the hidden weights

$$\vec{w} = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \end{bmatrix}, \qquad w^{(j)} \ge 0,\ \underbrace{w^{(1)} + w^{(2)} + w^{(3)}}_{=\mathbf{1}^{\top}\vec{w}} \;=\; 1,$$

assuming the meet really is using the linear rule $f(\vec{w};\vec{x}) = \vec{w}^{\top}\vec{x}$. Let

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \vec{w}^{\top}\vec{x}_i\big)^2 = \frac{1}{n}\,\|\vec{y} - \mathbf{U}\vec{w}\|^2, \qquad \mathbf{U} = \begin{bmatrix} \vec{x}_1^{\top} \\ \vdots \\ \vec{x}_n^{\top} \end{bmatrix} \in \mathbb{R}^{n\times 3}.$$

With $\mathbf{1} = (1,1,1)^{\top}$ and the sum-to-one constraint is $g(\vec{w}) = \mathbf{1}^{\top}\vec{w} - 1 = 0$. Introduce $\lambda \in \mathbb{R}$ and set

$$\mathcal{L}(\vec{w},\lambda) = R(\vec{w}) + \lambda\, g(\vec{w}) = \frac{1}{n}\,\|\vec{y} - \mathbf{U}\vec{w}\|^2 + \lambda\big(\mathbf{1}^{\top}\vec{w} - 1\big).$$

The **stationary conditions**, which are the constrained form of the normal equations, are given by

$$\nabla_{\vec{w}}\mathcal{L} = -\tfrac{2}{n}\mathbf{U}^{\top}\big(\vec{y} - \mathbf{U}\vec{w}\big) + \lambda\mathbf{1} = \vec{0}, \tag{25}$$

$$\partial\mathcal{L}/\partial\lambda = \mathbf{1}^{\top}\vec{w} - 1 = 0. \tag{26}$$

Assuming $\mathbf{U}^\top \mathbf{U}$ is invertible, (25) yields

$$\vec{w} = (\mathbf{U}^\top \mathbf{U})^{-1} \Big(\mathbf{U}^\top \vec{y} - \tfrac{n}{2} \lambda \mathbf{1} \Big). \tag{27}$$

Substituting (27) into (26) gives

$$\lambda^* = \frac{2}{n} \, \frac{\mathbf{1}^\top (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \vec{y} - 1}{\mathbf{1}^\top (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{1}}.$$

Finally,

$$\vec{w}^* = (\mathbf{U}^\top \mathbf{U})^{-1} \Big[\mathbf{U}^\top \vec{y} - \frac{\mathbf{1}^\top (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \vec{y} - 1}{\mathbf{1}^\top (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{1}} \, \mathbf{1} \Big] \tag{28}$$

which minimizes the mean-squared error while forcing the weights to sum to one. If desired, any negative entry of $\vec{w}^*$ can be interpreted as evidence that the published score is *not* a convex combination of the three panels, contradicting the coach's assumption.

3.5 Exercises

Exercise 3.1

Consider the following training dataset of scalar-valued input-output pairs:

$$\{(x_i, y_i)\}_{i=1}^{5} = \{(0, 1.2),\ (1, 2.8),\ (2, 4.5),\ (3, 7.1),\ (4, 8.9)\}.$$

Below are four candidate model formulas:

(i) Simple linear regression:

$$f(\vec{\theta};\, x) \;=\; \vec{\theta}^{(0)} + \vec{\theta}^{(1)}\, x.$$

(ii) Quadratic regression:

$$f(\vec{\theta};\, x) \;=\; \vec{\theta}^{(0)} + \vec{\theta}^{(1)}\, x + \vec{\theta}^{(2)}\, x^2.$$

(iii) Single exponential regression:

$$f(\vec{\theta};\, x) \;=\; \vec{\theta}^{(0)} + \vec{\theta}^{(1)}\, e^{-x}.$$

(iv) Multiple exponential regression:

$$f(\vec{\theta};\, x) \;=\; \vec{\theta}^{(0)} + \vec{\theta}^{(1)}\, e^{-x} + \vec{\theta}^{(2)}\, e^{-2x}.$$

And here are four design matrices built from the same x_i, with some entries rounded:

$$(A)\quad \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \quad (B)\quad \begin{bmatrix} 1 & 1.000 \\ 1 & 0.368 \\ 1 & 0.135 \\ 1 & 0.050 \\ 1 & 0.018 \end{bmatrix},$$

$$(C)\quad \begin{bmatrix} 1 & 1.000 & 1.000 \\ 1 & 0.368 & 0.135 \\ 1 & 0.135 & 0.018 \\ 1 & 0.050 & 0.002 \\ 1 & 0.018 & 0.000 \end{bmatrix}, \quad (D)\quad \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix}.$$

Match each model formula (i)-(iv) to its corresponding design matrix (A)-(D).

Exercise 3.2

In a cell proliferation assay, a biologist measures the cell density (in millions of cells per mL) at various time points (in hours) after seeding a petri dish with bacteria:

$$\{(x_i, y_i)\}_{i=1}^{8} = \{(1, 2.1),\ (2, 2.9),\ (3, 3.7),\ (4, 4.5),\ (5, 5.1),\ (6, 5.8),\ (7, 6.2),\ (8, 6.5)\}.$$

where x_i is time in hours and y_i is cell density. The biologist, looking to model this data, chooses to use simple polynomial regression with degree $d = 3$.

(a) For each point in the training dataset, write down the polynomial feature vector

$$[\vec{x_i}] = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \\ x_i^3 \end{bmatrix} \in \mathbb{R}^4,$$

and assemble the design matrix $[\mathbf{Z}] \in \mathbb{R}^{8\times 4}$.

(b) Using Theorem 3.1.1 and some `Python` where needed, compute the parameters $\vec{\theta}^*$ which minimize the MSE with respect to the training data.

(c) Produce a quick `Python` plot of the original data points and the fitted cubic curve over $x \in [0, 9]$.

Exercise 3.3

A climatologist measures ambient temperature y_i (in °C) at times x_i (in hours after sunrise) over a half-day:

$$\{(x_i, y_i)\}_{i=1}^{7} = \{(0, 15.0),\ (2, 17.3),\ (4, 20.1),\ (6, 22.8),\ (8, 21.5),\ (10, 18.2),\ (12, 16.0)\}.$$

To capture the periodic pattern, she considers the sinusoidal regression model

$$f(\vec{\theta};\, x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} \sin(0.1\, x) + \vec{\theta}^{(2)} \sin(0.2\, x) + \vec{\theta}^{(3)} \sin(0.3\, x).$$

(a) For each point in the training dataset, compute the transformed feature vector $[\vec{x_i}]$ and the corresponding design matrix $[\mathbf{Z}] \in \mathbb{R}^{7\times 4}$.

(b) Using Theorem 3.1.1 and some `Python` where needed, compute the parameters $\vec{\theta}^*$ which minimize the MSE with respect to the training data.

(c) Compute the training MSE of this sinusoidal model and compare it to the MSE of the simple linear model

$$f(\vec{\theta};\, x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} x.$$

(This requires a separate implementation of the simple linear model, which we have done plenty of times.) Which model is more appropriate here?

Exercise 3.4

Suppose we have training data $\{(x_i, y_i)\}_{i=1}^{n+1}$ with $x_i \in \mathbb{R}$ all distinct and $y_i \in \mathbb{R}$. Show that there exists a polynomial regression model of degree $d = n$ that fits the data with zero mean squared error:

$$f(\vec{\theta}; x) = \sum_{j=0}^{n} \vec{\theta}^{(j)} x^j, \qquad R(\vec{\theta}) = \frac{1}{n+1} \sum_{i=1}^{n+1} \left(y_i - f(\vec{\theta}; x_i)\right)^2 = 0.$$

(a) Show that the polynomial design matrix $[\mathbf{Z}]$ is invertible by using Theorem 3.1.2 and Exercise 2.16.

(b) Deduce that there is a unique solution $\vec{\theta}^*$ of the linear equation

$$[\mathbf{Z}]\, \vec{\theta}^* \;=\; \vec{y}$$

From this, complete the proof. (*Hint: it may help to rewrite the risk as a vector length.*)

Exercise 3.5

Below are four brief modeling scenarios. In each, the data scientist is considering adding pairwise interaction terms between the listed features. For each scenario, (i) list all possible interaction terms, and (ii) for each such term, speculate whether incorporating the interaction into a multiple linear regression model would be appropriate (clearly write "I" for include and "E" for exclude). You can base your answer on criteria such as whether or not the features operate completely independently from each other, or whether multiplying the features might present numerical issues related to their scale. *There will be more than one correct answer.*

(a) **(Housing prices)** Features: $x^{(1)}$ = square footage, $x^{(2)}$ = number of bedrooms, $x^{(3)}$ = age of house. Target: y = selling price of a home.

(b) **(Drug synergy)** Features: $x^{(1)}$ = dosage of Drug A, $x^{(2)}$ = dosage of Drug B, $x^{(3)}$ = patient weight. Target: y = survival time of mouse infected with

Disease X.

(c) **(Marketing effectiveness)** Features: $x^{(1)}$ = TV ad spend, $x^{(2)}$ = online ad spend, $x^{(3)}$ = seasonality index, $x^{(4)}$ = binary variable for new product that week. Target: y = weekly gross sales.

(d) **(Concrete fatigue)** Features: $x^{(1)}$ = temperature, $x^{(2)}$ = pressure, $x^{(3)}$ = humidity, $x^{(4)}$ = age of concrete. Target: y = weight the concrete slab can hold before breaking.

Exercise 3.6

In a neuroscience experiment, reaction time y_i (ms) is measured for stimulus intensity $x_i^{(1)}$ and attention level $x_i^{(2)}$ on $n = 6$ trials:

i	$x_i^{(1)}$	$x_i^{(2)}$	y_i
1	10	2	250
2	20	2	230
3	10	5	200
4	20	5	190
5	15	3	210
6	15	4	205

The neuroscientist assumes the model

$$f(\vec{\theta}, v, b; \vec{x}) = \vec{\theta}^{(1)} x^{(1)} + \vec{\theta}^{(2)} x^{(2)} + v\, x^{(1)} x^{(2)} + b.$$

(a) Construct the interaction design matrix

$$[\![\mathbf{Z}]\!] = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & x_1^{(1)} x_1^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_6^{(1)} & x_6^{(2)} & x_6^{(1)} x_6^{(2)} \end{bmatrix} \in \mathbb{R}^{6\times 4}, \quad \vec{y} = [y_1, \ldots, y_6]^\top.$$

(b) Using Theorem 3.1.4 and some `Python` where needed, compute the parameters $\vec{\theta}^*, v^*, b^*$ which minimize the MSE with respect to the training data. Then, compute the training MSE.

(c) Repeat this process without the interaction term $v\, x^{(1)} x^{(2)}$. Which model performs better? Is it worth the hassle to incorporate interaction here?

Exercise 3.7

Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be convex functions.

(a) Show that the sum $h(\vec{\theta}) = f(\vec{\theta}) + g(\vec{\theta})$ is convex.

(b) Suppose f is strictly convex and g is convex. Prove that $h(\vec{\theta}) = f(\vec{\theta}) + g(\vec{\theta})$ is strictly convex.

(c) Prove that any affine function $L(\vec{\theta}) = \vec{a}^\top \vec{\theta} + b$ is convex, where $\vec{a} \in \mathbb{R}^n, b \in \mathbb{R}$ are fixed. Is it strictly convex?

(d) Show that the pointwise maximum

$$m(\vec{\theta}) = \max\{f(\vec{\theta}), g(\vec{\theta})\}$$

is convex. *Hint: Use the epigraph definition of convexity, and consider some examples in the case of $n = 1$.*

(e) Let $a \geq 0$ be a fixed scalar. Prove that the nonnegative scaling $h(\vec{\theta}) = a\, f(\vec{\theta})$ is convex.

Exercise 3.8

Let $f : \mathbb{R} \to \mathbb{R}$ be twice differentiable. Prove that f is convex if and only if $f''(x) \geq 0$ for each $x \in \mathbb{R}$ by completing the following steps.

(a) Assume $f''(x) \geq 0$ for all x. Given $x < y$, apply the Mean Value Theorem to f on $[x, y]$ to find $c \in (x, y)$ with

$$f'(c) \;=\; \frac{f(y) - f(x)}{y - x}.$$

Then apply it again on $[x, c]$ and $[c, y]$ to show $f'(x) \leq f'(c) \leq f'(y)$.

(b) Deduce from (a) that for any $x < y$, it holds

$$f(y) - f(x) \;=\; \int_x^y f'(t)\, dt \;\geq\; \int_x^y f'(x)\, dt \;=\; f'(x)\,(y - x),$$

i.e. $f(y) \geq f(x) + f'(x)(y - x)$.

(c) For $x < y$ and $t \in [0, 1]$, set $z = (1 - t)x + ty$. Using the inequality in (b) twice (once between x and z, once between z and y) show that

$$f(z) \;\leq\; (1 - t)f(x) \;+\; t\, f(y).$$

Conclude that f is convex.

(d) Suppose f is convex. For any $x < y < z$, prove

$$\frac{f(y) - f(x)}{y - x} \;\leq\; \frac{f(z) - f(y)}{z - y}.$$

Fix x and let $y \to x^+$, $z \to x^-$. Show f' is monotone increasing. Conclude that $f''(x) \geq 0$ for all x.

Exercise 3.9

Determine for each of the following functions whether it is convex, strictly convex, or neither. Justify your answers.

(a) $f(x) = x^2 + 3x + 1$,

(b) $g(x) = |x|$,

(c) $h(x) = e^{-x}$,

(d) $\ell(\vec{x}) = \vec{x}^{(1)}\vec{x}^{(2)}, \quad \vec{x} \in \mathbb{R}^2$,

(e) $p(\vec{x}) = \|\vec{x}\|^2, \quad \vec{x} \in \mathbb{R}^n$.

Exercise 3.10

For each of the following objective functions, apply five iterations of the gradient descent algorithm (24) with the specified learning rate and initial point. Create a table with the iterates $\vec{\theta}_t$, the gradients $\nabla f(\vec{\theta}_t)$, and the corresponding function values $f(\vec{\theta}_t)$ (see Example 3.2.5 and Example 3.2.6 for examples of tables). Show ALL of your steps, you may ONLY use `Python` to carry out long calculations (like you would use a calculator).

(a) $f(\theta) = \theta^2, \quad \eta = 0.2, \quad \theta_0 = 2.$

(b) $f(\theta) = e^{\theta^2}, \quad \eta = 0.05, \quad \theta_0 = 1.5.$

(c) $f(\vec{\theta}) = \left(\vec{\theta}^{(1)}\right)^2 + 2\left(\vec{\theta}^{(2)}\right)^2, \quad \eta = 0.1, \quad \vec{\theta}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$

(d) $f(\vec{\theta}) = \left(\vec{\theta}^{(1)} - 1\right)^2 + \left(\vec{\theta}^{(2)} + 2\right)^2, \quad \eta = 0.15, \quad \vec{\theta}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$

Exercise 3.11

For each of the functions and hyperparameters below, perform five iterations of the finite-difference approximate gradient descent method (see Example 3.2.6 or Example 3.3.4), using step size $t_0 = 10^{-4}$ to approximate each partial derivative. Create a table with the iterates $\vec{\theta}_t$, the gradients $\nabla f(\vec{\theta}_t)$, and the corresponding function values $f(\vec{\theta}_t)$. You *can and should* use `Python` to carry out the computations algorithmically. You may generate the desired tables in `Python` and submit them as screenshots or replicate them by hand with a three-decimal rounding precision.

(a) $f(\theta) = |\theta|, \quad \eta = 0.1, \quad \theta_0 = 1.$

(b) $f(\theta) = \theta^4, \quad \eta = 10^{-3}, \quad \theta_0 = 1.$

(c) $f(\vec{\theta}) = |\vec{\theta}^{(1)}| + (\vec{\theta}^{(2)})^2, \quad \eta = 10^{-2}, \quad \vec{\theta}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$

(d) $f(\vec{\theta}) = \sin(\vec{\theta}^{(1)}) + \cos(\vec{\theta}^{(2)})$, $\eta = 5 \times 10^{-2}$, $\vec{\theta}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Exercise 3.12

Devise a simple real-world scenario in which you wish to predict a scalar target y from $d = 3$ features. Look up a real dataset or generate a synthetic dataset of $n = 10$ observations

$$\{(\vec{x}_i, y_i)\}_{i=1}^{10}, \qquad \vec{x}_i \in \mathbb{R}^3,\ y_i \in \mathbb{R}.$$

(*Do yourself a favor for what is to follow – keep the data model simple and sweet! Complicated data will make the next part harder for you.*) Then:

(a) Construct and store the design matrix for the corresponding multiple linear regression model $f(\vec{\theta};\, \vec{x})$ (interaction terms are left for you to decide!), as well as the label vector and any other ingredients needed for computing the risk.

(b) Using the MAE risk

$$R_{\text{abs}}(\vec{\theta}) = \frac{1}{10} \sum_{i=1}^{10} \left| y_i - f(\vec{\theta};\, \vec{x}_i) \right|,$$

approximate the gradient of R_{abs} by finite differences (step size $t_0 = 10^{-4}$) and implement gradient descent in `Python` to compute an approximate minimizers $\vec{\theta}^*$ of the MAE for the model $f(\vec{\theta};\, \vec{x})$. Use a suitably small learning rate (this may need to be quite small, e.g., $\eta \sim 10^{-7}$ or smaller) and enough iterations ($T \sim 10^5$ could be necessary) to ensure your approximate parameters are close to a true minimizer.

(c) Report the final model parameters and the training MAE. Generate a `Python` plot of the MAE risk vs. iteration number.

Exercise 3.13

Let $\{(x_i, y_i)\}_{i=1}^n$ be scalar-valued training data. Consider the simple linear regression model

$$f(\vec{\theta};\, x) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} x, \qquad \vec{\theta} = \begin{bmatrix} \vec{\theta}^{(0)} \\ \vec{\theta}^{(1)} \end{bmatrix} \in \mathbb{R}^2,$$

and the *ridge-regularised* mean-squared error

$$R_\mu(\vec{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f(\vec{\theta};\, x_i) \right)^2 + \mu (\vec{\theta}^{(1)})^2, \qquad \mu > 0.$$

Use Theorem 3.3.2 in the $d = 1$ setting to write down an analogue of Theorem 2.1.2 in for simple linear ridge regression.

Exercise 3.14

An ice-cream shop in Palm Springs records the average daily temperature x_i (in C) and the number of cones sold y_i (in hundreds) on eight summer days:

$$\{(x_i, y_i)\}_{i=1}^{8} = \{(18, 2.1), (20, 2.5), (22, 3.0), (24, 3.6), (26, 3.8), (28, 4.2), (30, 4.6), (32, 5.0)\}.$$

The manager, who is a retired data scientist, fits the simple linear model $f(\vec{\theta};\, x) = \vec{\theta}^{(1)} x + \vec{\theta}^{(0)}$ with an lasso penalty:

$$R_\mu(\vec{\theta}) = \frac{1}{8} \sum_{i=1}^{8} \big(y_i - f(\vec{\theta};\, x_i)\big)^2 + \mu |\vec{\theta}^{(1)}|, \qquad \mu \geq 0.$$

(a) Using finite-difference approximate gradients (step $t_0 = 10^{-4}$) and gradient descent, implement a `Python` method that returns $\vec{\theta}^{*}_{\mu}$ for $\mu \in \{0, 0.1, 1\}$. Use a suitably small learning rate (this may need to be quite small, e.g., $\eta \sim 10^{-7}$ or smaller) and enough iterations ($T \sim 10^5$ could be necessary) to ensure your approximate parameters are close to a true minimizer.

(b) Report the three pairs $(\vec{\theta}^{(1)*}_{\mu}, \vec{\theta}^{(0)*}_{\mu})$ and the corresponding training MSE. Comment briefly on how increasing μ alters the slope of the fitted line and speculate as to why this might be the case.

Exercise 3.15

A battery-testing lab measures the life-span y_i (in hours) of a smartphone under three operating conditions:

$$\begin{aligned} \vec{x}_i^{(1)} &= \text{screen brightness (\% of max)}, \\ \vec{x}_i^{(2)} &= \text{number of running apps}, \\ \vec{x}_i^{(3)} &= \text{device age (months)}. \end{aligned}$$

Ten experiments yield

i	$\vec{x}_i^{(1)}$	$\vec{x}_i^{(2)}$	$\vec{x}_i^{(3)}$	y_i
1	20	2	1	16.0
2	30	3	2	14.2
3	40	4	2	12.6
4	50	5	3	11.3
5	60	5	4	9.8
6	70	6	4	8.7
7	40	2	1	15.2
8	55	3	2	12.0
9	35	4	3	12.8
10	45	5	3	11.0

The lab models y with multiple linear regression

$$f(\vec{\theta};\, \vec{x}) = \vec{\theta}^{(0)} + \vec{\theta}^{(1)} \vec{x}^{(1)} + \vec{\theta}^{(2)} \vec{x}^{(2)} + \vec{\theta}^{(3)} \vec{x}^{(3)}, \qquad \vec{\theta} \in \mathbb{R}^4,$$

and employs the ridge-regularised mean absolute error

$$R_\mu(\vec{\theta}) = \frac{1}{10}\sum_{i=1}^{10}\left|y_i - f(\vec{\theta};\,\vec{x}_i)\right| + \mu\sum_{j=1}^{3}(\vec{\theta}^{(j)})^2, \qquad \mu > 0.$$

(a) Construct and store the design matrix for the corresponding multiple linear regression model $f(\vec{\theta};\,\vec{x})$ (interaction terms are left for you to decide!), as well as the label vector and any other ingredients needed for computing the risk.

(b) Using finite-difference approximate gradients (step $t_0 = 10^{-4}$) and gradient descent, implement a `Python` method that returns $\vec{\theta}^*_\mu$ for $\mu = 0.05$. Use a suitably small learning rate (this may need to be quite small, e.g., $\eta \sim 10^{-7}$ or smaller) and enough iterations ($T \sim 10^5$ could be necessary) to ensure your approximate parameters are close to a true minimizer.

(c) Report $\vec{\theta}^*$ and the achieved training MAE. Discuss qualitatively how the ridge term affects the weights compared with setting $\mu = 0$.

4 Modeling with Probability

It is remarkable that a science which began with the consideration of games of chance should have become the most important object of human knowledge.

- *Pierre-Simon Laplace*

In the earlier chapters of this book, we approached data science problems from the perspective of **deterministic models**, which are characterized by the working hypothesis that the output of the underlying system can be described by a fixed mapping $y = f(x)$, which of course may depend on some parameters or unknown quantities that we seek to capture. These sorts of models often work perfectly fine, but they provide the scientists no means of capturing uncertainty in the underlying data or resulting predictions. In other words, although we may freely report the loss or error associated with a given prediction for an input-output pair *in our training data*, it becomes impossible to report error associated with a predicted outcome without the foreknowledge of its true output value.

In contrast, a probabilistic approach treats the input and output variables as a random variables modeled via probability distributions, which are functions defined on the space of all possible outputs (e.g. real numbers, classification categories, etc.) and which reveals information about how likely particular outputs may be. This lets the scientist quantify confidence in predictions, incorporate prior knowledge through distributions, and gracefully handle missing or noisy observations.

Our main examle of a probabilistic model will be the Naïve Bayes classification model, which will be introduced in Section 4.4. The majority of this chapter will cover some fundamental concepts and tools from probability theory, which will be of great use in this course and beyond.

4.1 Sample Spaces and Probability Measures

A **sample space** Ω is the set of all possible outcomes of a random experiment. For this course we always assume Ω is a finite set. An **event** is any subset $A \subseteq \Omega$ corresponding to outcomes of interest. We denote by 2^Ω the collection of all subsets of Ω (including the empty set). A **probability measure** is a function $\mathbb{P}(\cdot) : 2^\Omega \to \mathbb{R}$ assigning to each event A a number $\mathbb{P}(A)$ satisfying the following three properties:[2]

(i) $\mathbb{P}(A) \geq 0$ for each $A \subseteq \Omega$,

(ii) $\mathbb{P}(\Omega) = 1$,

[2]Probability measures are a theoretical construct which allow us to model all of the intuitive beliefs and sometimes weird qualities humans associate with randomness. There is a long and rich history on this topic. See Wikipedia: History of probability.

(iii) If $A_1, A_2, \ldots, A_k \subseteq \Omega$ are events such that $A_i \cap A_j = \varnothing$ for each $1 \leq i, j \leq k$ with $i \neq j$, then $\mathbb{P}\left(\bigcup_{i=1}^{k} A_i\right) = \sum_{i=1}^{k} \mathbb{P}(A_i)$.

It follows that for any event $A \subseteq \Omega$, the complement $A^c = \{\omega \in \Omega : \omega \notin A\}$ satisfies $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$.

? You should prove this on your own.

In this course, we will focus on **discrete sample spaces**, which are sample spaces Ω that are either finite or can be enumerated in the form $\omega_1, \omega_2, \ldots, \ldots$ (e.g., "heads or tails," or "1, 2, 3, ..."). These are distinguished from **continuous sample spaces**, which are inifinite sample spaces that cannot be nicely counted (e.g., $\mathbb{R}$ or $\mathbb{R}^n$, etc.). Probability theory on these types of sample spaces require more technical machinery so we will save those for later study.

Key Idea

Probability measures can be specified in many different ways. In this course, when Ω is finite we will usually specify the measure by assigning to each outcome $\omega \in \Omega$ a probability $\mathbb{P}(\{\omega\})$. Afterwards, if $A \subseteq \Omega$ is any event, $\mathbb{P}(A)$ is simply the sum $\sum_{\omega \in A} \mathbb{P}(\{\omega\})$.

We distinguish a specific outcome $\omega \in \Omega$ from the event containing just that outcome, which is denoted $\{\omega\} \subseteq \Omega$. This is because probability measures are, strictly speaking, defined on events and not outcomes. In the discrete setting, this distinction can be dropped without many issues, but we will emphasize it here for the sake of precision.

Example 4.1.1

Consider rolling a fair six-sided die. The sample space is given by

$$\Omega = \{1, 2, 3, 4, 5, 6\},$$
$$\mathbb{P}(\{i\}) = \tfrac{1}{6} \quad \text{for each } i = 1, 2, \ldots, 6.$$

We can consider different example of events:

$$A = \{\text{roll an even number}\} = \{2, 4, 6\}$$
$$B = \{\text{roll at least a four}\} = \{4, 5, 6\}$$
$$C = \{\text{roll a two or a three}\} = \{2, 3\}$$

Then we have:

$$\mathbb{P}(A) = \sum_{i \in \{2,4,6\}} \mathbb{P}(\{i\}) = \frac{3}{6}$$
$$\mathbb{P}(B) = \sum_{i \in \{4,5,6\}} \mathbb{P}(\{i\}) = \frac{3}{6}$$
$$\mathbb{P}(C) = \sum_{i \in \{2,3\}} \mathbb{P}(\{i\}) = \frac{2}{6}.$$

Events should *always* be specified in such a manner so that once the random trial or experiment has been performed, it is completely unambiguous as to whether the event occurred or not. Suppose that Ω is as in Example 4.1.1.

VALID EVENTS	"the die rolls an even number," "the die rolls a multiple of three"
NOT EVENTS	"the die rolls a good number," "the die bounces two times"

The idea of "good numbers" are vague in this context, so the scientist should define these notions explicitly. On the other hand, one might certainly count the number of times the die bounces when tossed, but if this detail is not accounted for in the sample space, we cannot treat it as an event. In such case, we should re-define Ω. Note that in many instances, we might not always write down Ω explicitly (especially when it is very large). However, for one's first encounters with the concept, it is instructive to see it expressed explicitly as in the examples in this section.

🍊 Example 4.1.2

Suppose we are recording the weather on a given weekend. Each day can be recorded as either S (sunny), R (rainy), or C (cloudy). Therefore, the sample space consists of all pairs of outcomes (ω_1, ω_2) where $\omega_i \in \{S, R, C\}$ for $i = 1, 2$. The outcome (S, C) represents the observation that Saturday was sunny and Sunday was cloudy. Based on historical data, we can describe the probabilities of each outcome in the table below:

? Verify that the entries sum to 1.

Sat\Sun	S	R	C
S	0.25	0.15	0.10
R	0.15	0.09	0.06
C	0.10	0.06	0.04

Consider the two following events:

$$A = \{ \text{ no rainy days } \}$$
$$B = \{ \text{ at most one cloudy day} \}$$

We can compute their probabilities as follows.

$$\begin{aligned} \mathbb{P}(A) &= \mathbb{P}(\{(S,S),(S,C),(C,S),(C,C)\}) \\ &= 0.25 + 0.10 + 0.10 + 0.04 \\ &= 0.49, \\ \mathbb{P}(B) &= 1 - \mathbb{P}(\{\text{more than 1 cloudy day}\}^c) \\ &= 1 - \mathbb{P}(\{(C,C)\}) \\ &= 0.96. \end{aligned}$$

The second calculation is an example of what is sometimes called the **complement trick**, where computing the probability of the complement event and subtracting it from one is easier than computing the probability of the event itself.

A useful tool for computing probabilities is what we will call the **union rule**, which states that if $A, B \subseteq \Omega$ are any two events, then

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B).$$

Hopefully this statement is somewhat intuitive; we illustrate it in Fig. 8. We encourage the reader to write a proof on their own: refer to the proof of Theorem 4.3.4 later on if inspiration is needed.

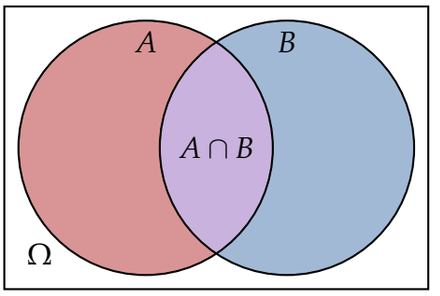


Figure 8: Visualizing the union rule with a Venn diagram.

The last example in this section is an important one which will start chipping away at the central application of this chapter.

Example 4.1.3 *Binary Feature Vectors*

Let $d \geq 1$ be a fixed integer. Then the **sample space of binary feature vectors** is given by

$$\Omega = \{\vec{x} \in \mathbb{R}^d : \vec{x}^{(i)} \in \{0,1\} \text{ for each } 1 \leq i \leq d\}.$$

For example, if $d = 3$, Ω consists of eight possible vectors, given below:

$$\Omega = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

More generally, Ω consists of 2^d elements. Note that we can specify any probability measure on Ω that we wish to consider; not simply the uniform one– this will be crucial later.

4.2 Counting and Combinatorics

Our next section concerns the following important question: how, in practice, do we compute probabilities? In many scenarios where the sample space is finite, this often simply reduces to counting the number of outcomes in a particular event. The logic underlying this observation is worth stating as a theorem below. If A is any set, we write $\#A$ to denote the cardinality of A, i.e., the number of elements in the set.

Theorem 4.2.1

Let Ω be a finite sample space. Let $\mathbb{P}(\cdot)$ be the uniform measure which associates an equal probability to each singleton event $\{\omega\} \subseteq \Omega$. Let $A \subseteq \Omega$ be any event.

Then

$$\mathbb{P}(A) = \frac{\#A}{\#\Omega}.$$

Proof

$$\mathbb{P}(A) = \sum_{\omega \in A} \mathbb{P}(\{\omega\}) = \sum_{\omega \in A} \frac{1}{\#\Omega} = \frac{\#A}{\#\Omega}.$$

This proof highlights a deeply intuitive property of probability: If all outcomes in a random experiment are equally likely, the probability of a particular event is just the ratio of the number of outcomes in the event to the number of all possible outcomes. Theorem 4.2.1 also suggests that whenever we need to compute probabilities in scenarios where the underlying measure is uniform, it amounts to counting the number of outcomes in both the event of interest and its parent sample space.

Combinatorics is the branch of mathematics that studies the counting, arrangement, and selection of discrete objects. For our purposes, we will cover a few basic concepts and techniques which concern counting certain arrangements and outcomes of processes. Sometimes the following are referred to as the "fundamental rules of counting;" the language is a bit grandiose since these rules will hopefully seem fairly obvious, but it is worth emphasizing their role in how we systematically count complex collections of outcomes.

Product Rule

Suppose a process (random or otherwise) occurs in k stages which are all independent of each other. If the i-th stage has $n_i \geq 1$ possibilities, then the total number of outcomes across all stages in the process is $n_1 \cdot n_2 \cdots n_k$.

A simple example: if one rolls a six-sided die 10 times, there are 6^{10} possible sequences of die rolls.

Sum Rule

Suppose a process (random or otherwise) has k sets of mutually exclusive outcomes. If the i-th set has n_i possibilities, then the total number of possible outcomes of the process is $n_1 + n_2 + \ldots + n_k$.

The sum rule applies when we classify outcomes into disjoint types (for instance, license plates might be of *Type A* or *Type B*). Counting each type separately and then summing leads to the desired total. The next two tools, permutations and combinations, are a bit richer.

Permutations

Suppose we begin with n distinct objects and we wish to count the number of arrangements of k of them **in order** (i.e., with emphasis on the first object, the second, and so on). Then there are

$$n(n-1)\cdots(n-k+1) = \frac{n!}{(n-k)!}$$

possible arrangements. The **factorial** $n!$ counts the number of ways to order all n objects and is defined by

$$0! = 1, \quad n! = n\,(n-1)! \quad \text{for } n \geq 1.$$

We use the notation $\mathsf{P}\,(n,k)$ to refer to the number of ordered arrangements of k items from a collection of n distinct items.

You have hopefully seen these numbers before: for example, $n!$ shows up in the Taylor series expansions of smooth functions in Calculus. $n!$ is notable for its rapid growth as n gets large, as evidenced below.

n	1	2	3	4	5	6	7	8	9	10
$n!$	1	2	6	24	120	720	5040	40320	362880	3628800

Combinations

Suppose we begin with n distinct objects and we wish to count the number of selections of k of them **without regard to order** (i.e., we only care about the k objects as a set). Then there are

$$\frac{n(n-1)\cdots(n-k+1)}{k!} = \frac{n!}{(n-k)!\,k!}$$

possible combinations. We use the notation $\mathsf{C}\,(n,k)$ to refer to the number of unordered combinations of k items from a collection of n distinct items.

$\mathsf{C}\,(n,k)$ is often also written in the notation $\binom{n}{k}$ and is more commonly read as "n choose k." These are also often referred to as the **binomial coefficients**, owing to the fact that for any $x, y \in \mathbb{R}$ and $n \geq 1$, one has

$$(x+y)^n = \sum_{k=0}^{n} x^k y^{n-k} \mathsf{C}\,(n,k)\,.$$

Some basic instances of combinatorial coefficients which are worth memorizing are given in the theorem below.

Theorem 4.2.2 *Theorem*

Let $n \geq 1$ be fixed, and $0 \leq k \leq n$. Then we have the following useful identities:

(i) $\mathsf{C}\,(n,0) = \mathsf{C}\,(n,n) = 1$,

(ii) $\mathsf{C}\,(n,1) = \mathsf{C}\,(n,n-1) = n$,

(iii) $\mathsf{C}(n,2) = \mathsf{C}(n,n-2) = \frac{n(n-1)}{2}$,

(iv) $\mathsf{C}(n,k) = \mathsf{C}(n,n-k)$.

Proof

Statement (i) can be seen by inspection: the number of ways to pick zero objects from a set of n of them is one; we just don't pick anything at all. Similarly, there is only one way to choose all n objects.

Statement (ii) falls along similar lines; there are n ways to pick exactly one object, and n ways to pick $n-1$ objects if we identify a selection of $n-1$ objects with the missing one.

Statement (iii) follows from the formula for combinatorial coefficients, but is worth memorizing nonetheless since it appears quite often.

Statement (iv) is similar to the latter case of statement (ii): if you count all of the ways to choose k objects from n, and for each such choice identify the complement of the corresponding set, you arrive at an enumeration of all ways of choosing $n-k$ objects from n.

With these ingredients alone, we can solve a wide array of counting problems; some of which can be quite challenging despite the simplicity and familiarity of the tools presented above.

Example 4.2.3 *Designing Secure Access Codes*

A research lab issues one-time "access codes" of the form `[L][D][S][S][N]`, where

- `[L]` is any of the 26 uppercase letters;
- `[D]` is one of the strings `AI`, `DS`, or `ML`;
- `[S]` is a *symbol* chosen from $\{\#, \$, !, @\}$, and the *two* `S` positions must be *different*;
- `[N]` is a three-digit number with no repeated digit.

Suppose we wish to find the probability that a randomly generated access code matches the pattern

`[vowel] DS #@ [even number]`.

The sample space Ω consists of all possible codes, so by Theorem 4.2.1, we can proceed in two steps by first counting all possible codes and then counting the number of codes which match the desired pattern. By the product rule,

$$\begin{aligned} \#\Omega &= 26 \times 3 \times \mathsf{P}(3,2) \times \mathsf{P}(10,3) \\ &= 26 \cdot 3 \cdot 4 \cdot 3 \cdot 720 \\ &= 673920. \end{aligned}$$

Let A be the event that the randomly generated code matches the desired pattern. Then again by the product rule,

$$\#A = 5 \times 1 \times 1 \times \#\{\text{even 3-digit numbers without repeating digits}\}.$$

To count the even numbers, notice that two criteria can be stated as: the 3-digits are all distinct, and the last digit is even. Since there are five different even digits $0, 2, 4, 6, 8$, by the sum rule, we can split all of the possibilities into five mutually exclusive sets corresponding to each digit. In each case, there are $\mathsf{P}(9,2)$ different ways to assign the leading two digits. Therefore,

$$\#\{\text{even 3-digit numbers without repeating digits}\} = 5 \times \mathsf{P}(9,2) = 5 \times 9 \times 8.$$

In conclusion,

$$\#A = 5 \times 5 \times 9 \times 8 = 1800$$

strings match our pattern. Therefore,

$$\mathbb{P}(A) = \frac{1800}{673920} \approx 0.0026.$$

So there is approximately a 1 in 375 chance of matching the pattern.

🍊 Example 4.2.4 *Counting cards*

A standard 52-card deck is thoroughly shuffled. We draw 7 cards at random. Suppose we wish to find the probability that we obtain exactly three hearts. The sample space has $\mathsf{C}(52,7)$ equally likely hands. To count the number of hands which have exactly three hearts, we note that there are $\mathsf{C}(13,3)$ ways to select the three hearts. Separately, there are $\mathsf{C}(39,4)$ ways to select the remaining four cards while excluding all hearts cards from the rest of the deck. Thus

$$\mathbb{P}(\{\text{hand contains 3 hearts}\}) = \frac{\mathsf{C}(13,3)\ \mathsf{C}(39,4)}{\mathsf{C}(52,7)} \approx 0.175.$$

🍊 Example 4.2.5 *Committee Selection*

A university Physics department has 8 men and 6 women. A committee of 5 people is chosen uniformly at random. Suppose we wish to compute the probability that the committee has at least 3 women. The sample space consists of

$$\#\Omega = \mathsf{C}(14,5)$$

equally likely committees. Let A denote the event that the committee contains at least 3 women. By the sum rule (summing over exactly 3, 4, or 5 women),

$$\begin{aligned}\#A = {} & \#\{\text{committees with three women}\} + \#\{\text{committees with four women}\} \\ & + \#\{\text{committees with five women}\}.\end{aligned}$$

The number of committees with exactly three women can be computed using the product rule: there are $\mathsf{C}(6,3)$ ways of choosing the three women, and $\mathsf{C}(8,2)$ ways of choosing the two men. Therefore,

$$\#\{\text{committees with three women}\} = \mathsf{C}(6,3) \times \mathsf{C}(8,2).$$

By extending this to the other two cases of four and five women, we conclude that

$$\#A = \mathsf{C}(6,3)\ \mathsf{C}(8,2) + \mathsf{C}(6,4)\ \mathsf{C}(8,1) + \mathsf{C}(6,5)\ \mathsf{C}(8,0).$$

Therefore by Theorem 4.2.1,

$$\begin{aligned}\mathbb{P}(\text{at least 3 women}) &= \frac{\mathsf{C}(6,3)\ \mathsf{C}(8,2) + \mathsf{C}(6,4)\ \mathsf{C}(8,1) + \mathsf{C}(6,5)\ \mathsf{C}(8,0)}{\mathsf{C}(14,5)} \\ &= \frac{686}{2002} \approx .343.\end{aligned}$$

4.3 Independence and Conditional Probability

In many experiments, we encounter events that can occur without influencing each other in any meaningful way. We say that two events A and B are **independent** if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\,\mathbb{P}(B).$$

? Show that if A and B are independent then A and B^c are also independent.

Intuitively, learning that B occurred does not change the probability of A. More generally, a collection of events $A_1, A_2, \ldots, A_n$ is **mutually independent** if every finite intersection factorizes into the product of its probabilities.

🍊 Example 4.3.1

Suppose we flip two fair coins. The sample space is

$$\Omega = \{(H,H),(H,T),(T,H),(T,T)\}, \quad \mathbb{P}(\{\omega\}) = \frac{1}{4} \text{ for each } \omega \in \Omega.$$

Let

$$\begin{aligned}A &= \{\text{first coin is heads}\} = \{(H,H),(H,T)\}, \\ B &= \{\text{second coin is heads}\} = \{(H,H),(T,H)\}.\end{aligned}$$

Then

$$\mathbb{P}(A) = \tfrac{1}{2}, \quad \mathbb{P}(B) = \tfrac{1}{2}, \quad \mathbb{P}(A \cap B) = \mathbb{P}(\{(H,H)\}) = \tfrac{1}{4},$$

so $\mathbb{P}(A \cap B) = \mathbb{P}(A)\,\mathbb{P}(B)$ and A, B are independent. Now let

$$C = \{\text{at least one head}\} = \{(H,H),(H,T),(T,H)\}.$$

Then

$$\begin{aligned}\mathbb{P}(C) &= \tfrac{3}{4},\\ \mathbb{P}(A \cap C) &= \mathbb{P}(A) = \tfrac{1}{2},\\ \mathbb{P}(A)\,\mathbb{P}(C) &= \tfrac{1}{2} \cdot \tfrac{3}{4} = \tfrac{3}{8} \neq \tfrac{1}{2}.\end{aligned}$$

Thus A and C are not independent.

Conditional probability allows us to compute probabilities of events when some additional (usually partial) information about the outcome is known. For events A and B with $\mathbb{P}(B) > 0$, the **conditional probability** of A *given* B is defined by

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{29}$$

This can be thought of as "the probability that A occurs if we know B has occurred." Alternatively, we can think of B as in some sense replacing the ground sample space Ω (think: the universe of possibilities shrinks to only those which are accounted for in B) and then $\mathbb{P}(A \mid B)$ amounts simply to computing the probability of the outcomes which are still possible (that is, $A \cap B$) and dividing it by the total probability of B to scale things appropriately.

Example 4.3.2

Suppose we draw two cards from a standard deck at random. Let

$$\begin{aligned}A &= \{\text{second card is an ace}\},\\ B &= \{\text{first card is an ace}\}.\end{aligned}$$

Note first that $\mathbb{P}(B) = \frac{4}{52}$, and that $\mathbb{P}(A \cap B) = \frac{\mathsf{C}(4,2)}{\mathsf{C}(52,2)} = \frac{4\times 3}{52\times 51}$, we have

$$\mathbb{P}(A \mid B) = \frac{\frac{4}{52}\frac{3}{51}}{\frac{4}{52}} = \frac{3}{51}.$$

Note that $\mathbb{P}(A) = \frac{4}{52}$ and $\mathbb{P}(A \mid B) = \frac{3}{51}$, so that $\mathbb{P}(A \mid B)$ is about 25% smaller. In other words, if we *know* that one ace has already been drawn, it is less likely that a second is also drawn compared to the probability of the second card being an ace on its own without knowledge of the first.

Example 4.3.3

Let Ω be a sample space and $A, B \subseteq \Omega$ any two events. Then A, B are independent if and only of $\mathbb{P}(A \mid B) = \mathbb{P}(A)$. This follows from the definition of conditional probability.

Example 4.3.3 reflects the intuitive idea that if events have outcomes that are completely independent of each other, knowledge of one's occurrence does not change the likelihood of the other.

Rearranging Eq. (29) leads to a particularly useful fact for computing probabilities:

$$\mathbb{P}(A \cap B) = \mathbb{P}(A \mid B)\,\mathbb{P}(B).$$

When an event A is difficult to analyze directly, we can "slice" the sample space into a convenient partition $\{B_i\}_{i=1}^k$ and evaluate A piece-by-piece within each slice. The **law of total probability** formalizes this idea, expressing $\mathbb{P}(A)$ as the sum of the smaller, and often easier-to-compute, probabilities $\mathbb{P}(A \cap B_i)$ (or $\mathbb{P}(A \mid B_i)\,\mathbb{P}(B_i)$) across the partition. We illustrate Theorem 4.3.4 in Fig. 9.

This is often called a **partition** of Ω and can be thought of as breaking Ω into several smaller pieces

Theorem 4.3.4 *Law of Total Probability*

Let Ω be a sample space, and suppose that $B_1, B_2, \ldots, B_k \subseteq \Omega$ are disjoint events such that $\bigcup_{i=1}^k B_i = \Omega$. Let $A \subseteq \Omega$ be any fixed event. Then the following two equations hold:

$$\mathbb{P}(A) = \sum_{i=1}^{k} \mathbb{P}(A \cap B_i)$$

$$\mathbb{P}(A) = \sum_{i=1}^{k} \mathbb{P}(A \mid B_i)\,\mathbb{P}(B_i).$$

The first equation is known as the intersection form of the law of total probability and the second is known as the conditional form of the law of total probability.

Proof

Since $\{B_i\}_{i=1}^k$ are disjoint and $\bigcup_{i=1}^k B_i = \Omega$, we can write

$$A = A \cap \Omega = A \cap \left(\bigcup_{i=1}^{k} B_i\right) = \bigcup_{i=1}^{k} (A \cap B_i),$$

where the unions are pairwise disjoint. Using the additivity of the probability measure,

$$\mathbb{P}(A) = \sum_{i=1}^{k} \mathbb{P}(A \cap B_i).$$

This is the intersection form. Next, for each i with $\mathbb{P}(B_i) > 0$, apply the definition of conditional probability:

$$\mathbb{P}(A \cap B_i) = \mathbb{P}(A \mid B_i)\ \mathbb{P}(B_i).$$

Substituting into the previous sum yields

$$\mathbb{P}(A) = \sum_{i=1}^{k} \mathbb{P}(A \cap B_i) = \sum_{i=1}^{k} \mathbb{P}(A \mid B_i)\,\mathbb{P}(B_i),$$

which is the conditional form.

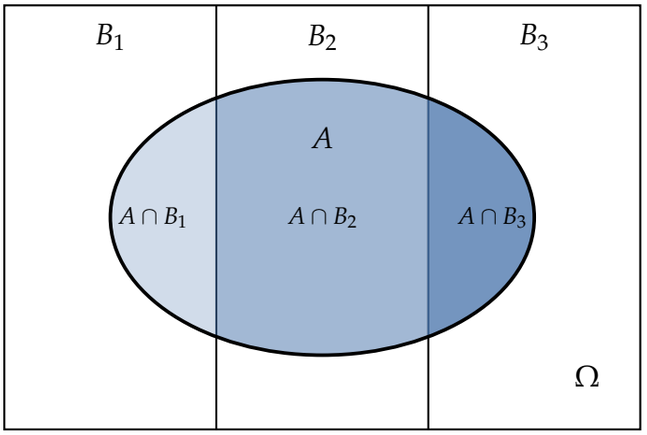


Figure 9: An illustration of the intersection form of the law of total probability in Theorem 4.3.4. Here, Ω is rendered as a rectangle which is partitioned into three events B_1, B_2, B_3. A generic event A is then split into three pieces based on its intersections with each slice. The probability of A is thus the sum of the probabilities of the pieces.

Bayes' theorem offers a principled way to *invert* conditional probabilities and compute these probabilities when, for example, knowledge of $\mathbb{P}(B \mid A)$ is readily found but $\mathbb{P}(A \mid B)$ is more difficult to compute.

Theorem 4.3.5 ***Bayes' Theorem***

Let Ω be a sample space and $A, B \subseteq \Omega$ any two events such that $\mathbb{P}(A), \mathbb{P}(B) > 0$. Then

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A)\,\mathbb{P}(A)}{\mathbb{P}(B)}.$$

Furthermore, we can write

$$\frac{\mathbb{P}(B \mid A)\,\mathbb{P}(A)}{\mathbb{P}(B)} = \frac{\mathbb{P}(B \mid A)\,\mathbb{P}(A)}{\mathbb{P}(B \mid A)\,\mathbb{P}(A) + \mathbb{P}(B \mid A^c)\,\mathbb{P}(A^c)}.$$

Proof

Recall that the conditional probability of A given B is

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Likewise, if $\mathbb{P}(A) > 0$,

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}.$$

Solving the latter for $\mathbb{P}(A \cap B)$ and substituting into the former gives

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A)\,\mathbb{P}(A)}{\mathbb{P}(B)},$$

which is Bayes' theorem. To obtain the alternative denominator, apply Theorem 4.3.4 to the event B with the partition $\{A, A^c\}$:

$$\mathbb{P}(B) = \mathbb{P}(B \cap A) + \mathbb{P}(B \cap A^c) = \mathbb{P}(B \mid A)\ \mathbb{P}(A) + \mathbb{P}(B \mid A^c)\,\mathbb{P}(A^c).$$

Substituting this expression for $\mathbb{P}(B)$ in the denominator above yields

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A)\ \mathbb{P}(A)}{\mathbb{P}(B \mid A)\,\mathbb{P}(A) + \mathbb{P}(B \mid A^c)\,\mathbb{P}(A^c)}.$$

Example 4.3.6 *Disease Testing*

A disease affects 1% of a population. A test for the disease is 95% accurate in both directions: that is, if a person is selected at random, then

$$\mathbb{P}(\text{test is positive} \mid \text{person has the disease}) = 0.95$$
$$\mathbb{P}(\text{test is negative} \mid \text{person is healthy}) = 0.95.$$

Suppose we wish to compute the probability that, if a person is selected at random and tests positive, they actually have the disease. Let

$$D = \{\text{patient has disease}\},$$
$$T = \{\text{test is positive}\}.$$

Then $\mathbb{P}(T \mid D) = 0.95$, $\mathbb{P}(D) = 0.01$, and

$$\mathbb{P}(T) = \mathbb{P}(T \mid D)\,\mathbb{P}(D) + \mathbb{P}(T \mid D^c)\,\mathbb{P}(D^c) = 0.95 \cdot 0.01 + 0.05 \cdot 0.99 = 0.059.$$

By Theorem 4.3.5,

$$\mathbb{P}(D \mid T) = \frac{0.95 \cdot 0.01}{0.059} \approx 0.161.$$

Thus even a positive test yields only about a 16% chance of disease.

This illustrates how false positives dominate when the condition is rare.

Example 4.3.7 *Example*

Robert has a sock drawer containing exactly eight socks: one pair each of plain socks, dotted socks, striped socks, and plaid socks. On Monday morning while getting ready, Robert selects two socks at random from the drawer. Suppose we wish to find the probability that the socks are matching. Since there are $\mathsf{C}(8,2)$ ways to pick the two socks from the drawer, of which exactly four pairs are matching. Therefore the probability is

$$\frac{4}{\mathsf{C}(8,2)} = \frac{(4)(2)}{(8)(7)} = \frac{1}{7}$$

Now suppose that after wearing the socks on Monday he places them in a laundry basket separate from the drawer. On Tuesday morning, as Robert gets ready, he selects another two socks at random from the six remaining in the drawer. Suppose

we wish to find the probability that the socks are matching on Tuesday: the key is that we can condition on whether the socks were matching on Monday and use this information to our advantage. Specifically,

$$\begin{aligned}\mathbb{P}(\text{match on Tues.}) &= \mathbb{P}(\text{match on Tues.}|\text{match on Mon.})\,\mathbb{P}(\text{match on Mon.}) \\ &\quad + \mathbb{P}(\text{match on Tues.}|\text{don't match on Mon.})\,\mathbb{P}(\text{don't match on Mon.}) \\ &= \mathbb{P}(\text{match on Tues.}|\text{match on Mon.})\,\frac{1}{7} \\ &\quad + \mathbb{P}(\text{match on Tues.}|\text{don't match on Mon.})\,(1-\frac{1}{7})\end{aligned}$$

If the socks were matching on Monday, there are now three pairs of matching socks in the drawer. So we have, by the same logic as before,

$$\mathbb{P}(\text{match on Tues.}|\text{match on Mon.}) = \frac{3}{\mathsf{C}(6,2)} = \frac{(3)(2)}{(6)(5)} = \frac{1}{5}.$$

If the socks were not matching on Monday, there are now two pairs of matching socks plus two mismatching socks. The total number of pairs is still $\mathsf{C}(6,2)$ but now only two such pairs match. Therefore,

$$\mathbb{P}(\text{match on Tues.}|\text{don't match on Mon.}) = \frac{2}{\mathsf{C}(6,2)} = \frac{(2)(2)}{(6)(5)} = \frac{2}{15}. \tag{30}$$

Thus in conclusion we have

$$\mathbb{P}(\text{match on Tues.}) = \frac{1}{5}\times\frac{1}{7} + \frac{2}{15}\times\frac{6}{7}. \tag{31}$$

Finally, we often need to consider events that are independent only after conditioning on some background information. Events A and B are **conditionally independent** given C when

$$\mathbb{P}(A\cap B \mid C) = \mathbb{P}(A \mid C)\,\mathbb{P}(B \mid C),$$

provided $\mathbb{P}(C) > 0$. This notion will be useful when we study Naïve Bayes classifiers, which will be covered in the next section.

Example 4.3.8 *Chess Tournament*

A chess player is entering a weekend tournament and is about to play two games on her first day. She is either having a good day (denoted "G"), or a bad day (denoted "B"); and she is more likely to win if she's having a good day. The sample space of outcomes corresponding to her two games is given by

$$\begin{aligned}\Omega = \{&(G,W,W),(B,W,W),(G,L,W),(B,L,W),\\ &(G,W,L),(B,W,L),(G,L,L),(B,L,L)\},\end{aligned}$$

where "W" denotes a win and "L" denotes a loss. For example, (B,W,L) denotes the outcome where she has a bad day, wins the first game, and loses the second.

On a typical day, the probability she has a good or a bad day is given by

$$P(\{\text{Good day}\}) = \mathbb{P}\left(\{(G,W,W),(G,L,W),\dots\}\right) = 0.6,$$
$$P(\{\text{Bad day}\}) = \mathbb{P}\left(\{(B,W,W),(B,L,W),\dots\}\right) = 0.4.$$

Let $A, B \subseteq \Omega$ be the events given by

$$A = \{\text{she wins in round 1}\}, \qquad B = \{\text{she wins in round 2}\}.$$

Suppose that *conditional on* the player's form, the outcomes of the two rounds are independent, with

$$P(A \mid \{\text{Good day}\}) = P(B \mid \{\text{Good day}\}) = 0.7,$$
$$P(A \mid \{\text{Bad day}\}) = P(B \mid \{\text{Bad day}\}) = 0.4.$$

Now suppose we wish to compute the following probabilities:

$$\mathbb{P}(A), \quad \mathbb{P}(B), \quad \mathbb{P}(A \cap B).$$

and determine whether A and B are independent unconditionally. By the *conditional* independence assumption,

$$\mathbb{P}(A \cap B \mid \{\text{Good day}\}) = 0.7 \cdot 0.7 = 0.49,$$
$$\mathbb{P}(A \cap B \mid \{\text{Bad day}\}) = 0.4 \cdot 0.4 = 0.16.$$

Hence by the law of total probability,

$$\mathbb{P}(A) = 0.6 \cdot 0.7 + 0.4 \cdot 0.4 = 0.42 + 0.16 = 0.58,$$
$$\mathbb{P}(B) = 0.6 \cdot 0.7 + 0.4 \cdot 0.4 = 0.58,$$
$$\mathbb{P}(A \cap B) = 0.6 \cdot 0.49 + 0.4 \cdot 0.16 = 0.294 + 0.064 = 0.358.$$

Since

$$\mathbb{P}(A)\,\mathbb{P}(B) = 0.58^2 = 0.3364 \neq 0.358 = \mathbb{P}(A \cap B),$$

the events A and B are not independent (even though they are conditionally independent given the form of the day).

4.4 Naïve Bayes Classifiers

An extremely common data science problem is that of **data classification**: this occurs when the target associated with a given input is a categorical variable (e.g. 'cat' / 'dog', or a digit between zero and nine). Naïve Bayes is a simple and effective approach to the data classification problem. The focus of this section will be to introduce and develop this model within the framework of the modeling method in a systematic fashion; doing so will take some careful setup.

For the most part, our philosophy of the modeling method carries over gracefully to the probabilistic paradigm. We restate it here with some changes to reflect this viewpoint.

The Modeling Method for Probabilistic Classifiers

1. Identify the sample space of input and output variables Ω.
2. Use your training observations to design a suitable probability measure on Ω.
3. Form predictions by computing $\mathbb{P}(y \mid x)$ for each possible target y.

Consider the following problem as a motivating example: we have a dataset of emails, and we will use it to train a model to predict whether a given email is legitimate or spam. We have a list of four keywords: `free`, `win`, `project`, and `meeting`; and for each of the six emails in our dataset and each word, we record whether the word is present in the text (represented by a 1) or not present in the text (represented by a zero.). The dataset is shown in Fig. 10 and Fig. 11.

i	1	2	3	4
word	`free`	`win`	`project`	`meeting`

Figure 10: The words cataloged in our spam email example.

email i	$\vec{x}_i^{(1)}$	$\vec{x}_i^{(2)}$	$\vec{x}_i^{(3)}$	$\vec{x}_i^{(4)}$	label	y_i
1	1	1	0	1	spam	1
2	1	0	0	0	spam	1
3	0	1	0	0	spam	1
4	0	0	1	1	not spam	0
5	1	0	1	0	not spam	0
6	0	0	0	1	not spam	0

Figure 11: The dataset of emails used to train our classification model.

In this case, we have four input features corresponding to the different keywords in our dictionary. For example, the fourth email in our dataset $\vec{x}_4$ contains the words "project" and "meeting" but neither of the words "free" or "win."

In this setting, model the features $\vec{x}$ as random observations from our probability space; in particular they are sampled from the sample space of *binary feature vectors*. The probability measure itself is to bed determined. We also model the labels y as random observations sampled from the collection of all possible labels $\{0, 1\}$ with, again, some unknown probability measure.

Keep in mind as you read on: our job is now to figure out how to pick the "best" probability measure on the sample space of feature-label pairs.

Denote the corresponding sample space of all *possible* input-output pairs by Ω. If $\mathcal{X}$ is the set of all d-dimensional binary feature vectors and $\mathcal{Y} = \{1, 2, \ldots, m\}$ is a generic set of m symbolic labels, then we can choose Ω to be the Cartesian product $\Omega = \mathcal{X} \times \mathcal{Y}$.

Key Idea

Our main goal in this modeling task is to specify, for a given feature vector $\vec{x}$, the *probabilities* that the example belongs to each class under consideration. Let Y denote the random label of the example (e.g., spam vs. not spam), and let $\vec{X}$ denote the feature vector, which we model as being random from among the d-dimensional binary feature vectors. With this setup, we are looking to find the "best" choice for a probability mesure, giving rise to the predicted probabilities

$$\mathbb{P}\left(\{Y = y\} \mid \{\vec{X} = \vec{x}\}\right), \qquad y \in \{1, 2, \ldots, m\}.$$

It might help to keep this task front of mind.

The key ingredient in this particular model is the **naïve Bayes assumption**: we assume that features are conditionally independent *given the class information of a target example*. In other words, if we know that an email is spam, the event that the email contains the word "free" is independent of the event that the email contains the word "win."

Let $(\vec{x}, y) \in \Omega$ be a specific outcome in our sample space (for example, a spam email containing `free` and `project`) and let $(\vec{X}, Y)$ be a random observation. The naïve Bayes assumption is that our model's probability measure satisfies

$$\mathbb{P}\left(\{\vec{X} = \vec{x}\} \mid \{Y = y\}\right) = \prod_{j=1}^{d} \mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} \mid \{Y = y\}\right). \tag{32}$$

Here, $\{\vec{X} = \vec{x}\}$ is the event that we observe a specific outcome $\vec{x}$ in our feature space. The event $\{\vec{X}^{(j)} = \vec{x}^{(j)}\}$ is the slightly more general event that we observe a feature vector which has the same feature in a specific coordinate. Similarly, $\{Y = y\}$ is the event that we observe an example that has label y.

As mentioned above, what we really want in our model is actually the reverse: ideally, our probabilistic model should give us a way to find

$$\mathbb{P}\left(\{Y = y\} \mid \{\vec{X} = \vec{x}\}\right),$$

since in practice we treat the observed features like an input to the trained model. By Bayes' theorem (Theorem 4.3.5) and Eq. (32), we can swap around the conditional probabilities to write our probabilistic model in the form

$$\begin{aligned}
&\mathbb{P}\left(\{Y = y\} \mid \{\vec{X} = \vec{x}\}\right) \\
&\qquad = \frac{\mathbb{P}\left(\{\vec{X} = \vec{x}\} \mid \{Y = y\}\right) \mathbb{P}\left(\{Y = y\}\right)}{\mathbb{P}\left(\{\vec{X} = \vec{x}\}\right)} \\
&\qquad = \frac{\prod_{j=1}^{d} \mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} \mid \{Y = y\}\right) \mathbb{P}\left(\{Y = y\}\right)}{\mathbb{P}\left(\{\vec{X} = \vec{x}\}\right)} \\
&\qquad \propto \prod_{j=1}^{d} \mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} \mid \{Y = y\}\right) \mathbb{P}\left(\{Y = y\}\right).
\end{aligned} \tag{33}$$

The quantity on the left-hand side is what we're looking for in a probabilistic model: for each class $y \in \mathcal{Y}$, it describes the probability that our input belongs to this class assuming we observe a specific set of input features. In the last line, we ignore the denominator $\mathbb{P}(\{\text{we observe } \vec{x}\})$, since it will be the same for each class y, and will therefore become redundant when we compute all of the class probabilities; we can simply divide each term by the total sum to obtain probabilities.

Our next step is to identify how we should approach computing the term

$$\mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} \mid \{Y = y\}\right). \tag{34}$$

To do this, we make use of our training data. We can estimate the probability in Eq. (34) as follows: among all training examples with class label y, what proportion of examples match the input feature $\vec{x}^{(j)}$? This computation replaces the standard optimization step; so if we can do this for each input feature $\vec{x}$ and class y, we have specified the predicted class distribution for our example.

Keeping the spam email dataset in mind, suppose we have a new input email with feature vector

$$\vec{x} = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^\top$$

indicating that the email contains the words `free` and `meeting`, but not the words `win` or `project`. We wish to predict whether this email is spam ($y = 1$) or not spam ($y = 0$). To apply the Naïve Bayes classification model, we compute the probabilities:

$$\mathbb{P}\left(\{Y = 1\} \mid \{\vec{X} = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^\top\}\right), \quad \mathbb{P}\left(\{Y = 0\} \mid \{\vec{X} = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^\top\}\right).$$

Using Eq. (33), we have for $y \in \{0, 1\}$,

$$\mathbb{P}\left(\{Y = y\} \mid \{\vec{X} = \vec{x}\}\right) \propto \mathbb{P}(\{Y = y\}) \prod_{j=1}^{4} \mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} \mid \{Y = y\}\right).$$

From our training data in Fig. 11, we estimate these probabilities separately for the two classes.

- **Class $y = 1$, spam:** From Fig. 11, we have 3 spam emails (emails 1, 2, and 3). Therefore we first have that

  $$\mathbb{P}(\{Y = 1\}) = \frac{3}{6} = \frac{1}{2},$$

  and given this, we can estimate the conditional probability

  $$\mathbb{P}\left(\{\vec{X}^{(1)} = 1\} \mid \{Y = 1\}\right) = \frac{2}{3},$$

  since two spam emails out of three contain the word `free`. Continuing in this manner,

  $$\mathbb{P}\left(\{\vec{X}^{(2)} = 0\} \mid \{Y = 1\}\right) = \frac{1}{3},$$
  $$\mathbb{P}\left(\{\vec{X}^{(3)} = 0\} \mid \{Y = 1\}\right) = \frac{3}{3} = 1$$
  $$\mathbb{P}\left(\{\vec{X}^{(4)} = 1\} \mid \{Y = 1\}\right) = \frac{1}{3}.$$

  Thus,

  $$\mathbb{P}\left(\{Y = 1\} \mid \{\vec{X} = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^\top\}\right) \propto \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot 1 \cdot \frac{1}{3} = \frac{1}{27}.$$

- **Class $y = 0$, not spam:** Similarly, from Fig. 11, we have 3 non-spam emails. We estimate

$$\begin{aligned}
\mathbb{P}\left(\{Y = 0\}\right) &= \frac{3}{6} = \frac{1}{2}, \\
\mathbb{P}\left(\{\vec{X}^{(1)} = 1\} \mid \{Y = 0\}\right) &= \frac{1}{3}, \\
\mathbb{P}\left(\{\vec{X}^{(2)} = 0\} \mid \{Y = 0\}\right) &= \frac{3}{3} = 1, \\
\mathbb{P}\left(\{\vec{X}^{(3)} = 0\} \mid \{Y = 0\}\right) &= \frac{2}{3}, \\
\mathbb{P}\left(\{\vec{X}^{(4)} = 1\} \mid \{Y = 0\}\right) &= \frac{2}{3}.
\end{aligned}$$

  Hence,

$$\mathbb{P}\left(\{Y = 0\} \mid \{\vec{X} = (1,0,0,1)\}\right) \propto \frac{1}{2} \cdot \frac{1}{3} \cdot 1 \cdot \frac{2}{3} \cdot \frac{2}{3} = \frac{2}{27}.$$

To find suitably normalized probabilities, divide each by their sum:

$$\begin{aligned}
\mathbb{P}\left(\{Y = 1\} \mid \{\vec{X} = (1,0,0,1)\}\right) &= \frac{1/27}{1/27 + 2/27} = \frac{1}{3} \approx 0.333, \\
\mathbb{P}\left(\{Y = 0\} \mid \{\vec{X} = (1,0,0,1)\}\right) &= \frac{2/27}{1/27 + 2/27} = \frac{2}{3} \approx 0.667.
\end{aligned}$$

Therefore, the Naïve Bayes classifier predicts that this email is *not spam*, with approximately a 66.7% probability.

🍊 Example 4.4.1 *Laplace Smoothing for News-Headline Classification*

In practice the *naïve Bayes* product in (33) can collapse to 0 whenever a particular feature never appears for one of the classes in the training set. This is sometimes called the **zero-frequency problem**. A technique called **Laplace smoothing** (also called *add-one* smoothing) fixes the issue by pretending that we have observed each possible outcome one extra time. Formally, for a sample $(\vec{x}, y)$ we replace the empirical estimate

$$\mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} | \{Y = y\}\right) = \frac{\#\{\text{samples in class } y \text{ with matching feature} \vec{x}^{(j)}\}}{\#\{\text{samples in class } y\}}$$

by the *smoothed* estimate

$$\mathbb{P}\left(\{\vec{X}^{(j)} = \vec{x}^{(j)}\} | \{Y = y\}\right) = \frac{\#\{\text{samples in class } y \text{ with matching feature} \vec{x}^{(j)}\} + 1}{\#\{\text{samples in class } y\} + 2}$$

The $+2$ comes from adding one pseudo-count to *both* of the two possible feature values 0 and 1. To see this in action, suppose we collect 20 short news headlines and label each as *Politics* ($y = 0$) or *Sports* ($y = 1$). For every headline we record five binary features indicating whether the words `goal`, `coach`, `election`, `policy`, `budget` appear. The dataset is shown below:

headline i	goal	coach	election	policy	budget	y
1	0	0	1	1	1	0
2	0	1	1	1	0	0
3	0	0	1	1	1	0
4	1	1	0	0	0	1
5	1	1	0	0	0	1
6	0	1	0	0	0	1
7	1	1	0	0	0	1
8	1	0	0	0	0	1

Next, consider the headline:

> "*Goal tally dominates policy debate ahead of election*"

whose feature vector is

$$\vec{x} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}^\top,$$

indicating that the words `goal`, `election`, and `policy` are present. *Without smoothing*, given the empirical probabilities above,

$$\mathbb{P}\left(\{X^{(1)} = 1 \mid Y = 0\}\right) = 0, \qquad \mathbb{P}\left(\{X^{(3)} = 1 \mid Y = 1\}\right) = 0,$$

so *both* class-likelihood products become 0 and the classifier fails to make a meaningful prediction. Using the smoothed estimate, we obtain

$$\begin{aligned} \mathbb{P}\left(\{Y = 0 \mid \vec{X} = \vec{x}\}\right) &\propto \frac{8}{20}\left(\tfrac{0+1}{8+2}\right)\left(\tfrac{1+1}{8+2}\right)^0\left(\tfrac{8+1}{8+2}\right)\left(\tfrac{5+1}{8+2}\right)\left(\tfrac{(8-5)+1}{8+2}\right) \\ &= 0.006912, \\ \mathbb{P}\left(\{Y = 1 \mid \vec{X} = \vec{x}\}\right) &\propto \frac{12}{20}\left(\tfrac{10+1}{12+2}\right)\left(\tfrac{7+1}{12+2}\right)^0\left(\tfrac{0+1}{12+2}\right)\left(\tfrac{0+1}{12+2}\right)\left(\tfrac{(12-0)+1}{12+2}\right) \\ &= 0.000957. \end{aligned}$$

Normalizing by their sum yields

$$\mathbb{P}\left(\{Y = 0 \mid \vec{X} = \vec{x}\}\right) \approx 0.88, \qquad \mathbb{P}\left(\{Y = 1 \mid \vec{X} = \vec{x}\}\right) \approx 0.12.$$

With Laplace smoothing, the classifier now makes a sensible prediction (*Politics*) instead of being stymied by zero probabilities.

4.5 Exercises

Exercise 4.1

Consider tossing *two* fair six-sided dice, one *red* and one *blue*.

(a) Write the sample space Ω in set-builder notation and state its cardinality.

(b) Let

$$A = \{\text{the sum of the two dice is } 7\}$$
$$B = \{\text{at least one die shows a } 5\}$$

Write the events A and B explicitly as a collection of events in Ω. Then compute $\mathbb{P}(A)$, $\mathbb{P}(B)$, and $\mathbb{P}(A \cup B)$.

(c) Verify the union rule for the events A and B computed in part (b).

Exercise 4.2

Fix $d \geq 1$ and recall the sample space of binary feature vectors $\Omega = \{0,1\}^d$ introduced in Example 4.1.3. Suppose we impose the *non-uniform* probability measure

$$\mathbb{P}(\{\vec{x}\}) = \frac{\lambda^{|\vec{x}|}}{(1+\lambda)^d}, \qquad \text{with } \lambda > 0,$$

where $|\vec{x}|$ counts the number of 1's in $\vec{x}$.

(a) Show that the above assignment is a valid probability measure on Ω.

(b) For fixed d and λ, compute the probability that a random vector has exactly k ones.

(c) When $\lambda = 1$ the measure is uniform. What value of λ makes the vector twice as likely to have a 1 in any given coordinate as a 0?

Exercise 4.3

A country issues licence plates of the form `AAA-NNN`, where each `A` is an uppercase letter excluding `O`, `I`, and `Q`, and each `N` is a digit excluding the possibility of a leading 0.

(a) How many distinct plates are possible?

(b) If a plate is chosen uniformly at random, what is the probability that all three letters are vowels?

(c) What is the probability that the three-digit number is a strictly increasing sequence (e.g. `135`)?

(d) Combine the previous answers to find the probability that a random plate has *both* properties.

Exercise 4.4

A department has 9 professors, 5 graduate students, and 4 staff members. A committee of 6 people is formed uniformly at random.

(a) In how many ways can the committee be formed?

(b) What is the probability the committee contains exactly 2 professors, 3 graduate students, and 1 staff member?

(c) What is the probability that the committee contains at least one person from each group?

Exercise 4.5

From a standard 52-card deck you draw 5 cards at random.

(a) Compute the probability of getting a *full house* (three of one rank and two of another).

(b) Compute the probability of getting a *flush* (all five cards are the same suit, ranks arbitrary).

(c) Which hand is more likely, a full house or a flush? Support your answer with calculations.

Exercise 4.6

A fair coin is flipped *three* times. Let the sample space be

$$\Omega = \{H, T\}^3.$$

Define the following events:

$$\begin{aligned} A &= \{\text{first flip is } H\}, \\ B &= \{\text{exactly two heads appear}\}, \\ C &= \{\text{third flip is } T\}. \end{aligned}$$

(a) Write each event as an explicit subset of Ω.

(b) Compute $\mathbb{P}(A)$, $\mathbb{P}(B)$, $\mathbb{P}(C)$, and all pairwise intersections.

(c) Determine which pairs among $\{A, B, C\}$ are independent.

(d) Are the three events mutually independent? Justify your answer.

Exercise 4.7

A student gets to campus by deciding at random each day from among one of three modes:

$$\begin{aligned} M_1 &= \{\text{bus}\}, \\ M_2 &= \{\text{bicycle}\}, \\ M_3 &= \{\text{car}\}, \end{aligned}$$

with

$$\begin{aligned}\mathbb{P}(M_1) &= 0.5,\\ \mathbb{P}(M_2) &= 0.3,\\ \mathbb{P}(M_3) &= 0.2.\end{aligned}$$

Let $L =$ "the student arrives *late*." Conditional probabilities collected over the semester are

$$\begin{aligned}\mathbb{P}(L \mid M_1) &= 0.15,\\ \mathbb{P}(L \mid M_2) &= 0.05,\\ \mathbb{P}(L \mid M_3) &= 0.10.\end{aligned}$$

(a) Use the law of total probability to compute $\mathbb{P}(L)$.

(b) Compute the probability that she came by bus given that she arrived late.

Exercise 4.8

A factory produces light bulbs. Each bulb first passes an automated test and then a manual inspection if it failed the first. After selecting a light bulb at random, let

$$\begin{aligned}A &= \{\text{bulb ultimately passes quality control}\},\\ T &= \{\text{bulb passes the automated test}\}.\end{aligned}$$

Historical records show that

$$\begin{aligned}\mathbb{P}(T) &= 0.92,\\ \mathbb{P}(A \mid T) &= 1,\\ \mathbb{P}(A \mid T^c) &= 0.40.\end{aligned}$$

(a) Verify that $\{T, T^c\}$ is a partition of Ω and compute $\mathbb{P}(A)$.

(b) Compute $\mathbb{P}(T \mid A)$, the probability a randomly selected *approved* bulb passed the automated test.

(c) Management claims "more than 95% of approved bulbs passed only the automated stage." Is the claim supported? Explain briefly.

Exercise 4.9

A simple spam filter flags an email as suspicious if it contains the word `urgent`. For a randomly selected email, database records suggest that

$$\begin{aligned}\mathbb{P}(\text{email is spam}) &= 0.12,\\ \mathbb{P}(\texttt{urgent} \text{ appears} \mid \text{spam}) &= 0.30,\\ \mathbb{P}(\texttt{urgent} \text{ appears} \mid \text{not spam}) &= 0.04.\end{aligned}$$

(a) Compute the probability that a random email contains the word `urgent`.

(b) Given that an email *does* contain `urgent`, use Bayes' theorem to compute the posterior probability it is spam.

(c) The filter quarantines every message with `urgent`. What fraction of quarantined emails are *actually not spam*? Comment on the effectiveness of this rule.

Exercise 4.10

A new screening test detects a rare genetic trait found in 0.7% of the population. Clinical trials report:

$$\begin{aligned}\mathbb{P}\left(\text{positive} \mid \text{trait}\right) &= 0.98,\\ \mathbb{P}\left(\text{negative} \mid \text{no trait}\right) &= 0.93.\end{aligned}$$

(a) If a randomly chosen individual tests positive, what is the probability they actually carry the trait?

(b) If the same individual tests negative, what is the probability they are trait-free?

(c) Discuss—in two or three sentences—whether this test is more reliable for ruling the presence of the trait *in* or *out*.

Exercise 4.11

A rare disease D affects 1% of the population. A randomly selected individual is monitored in a clinic for the appearance of two symptoms:

$$\begin{aligned}S_1 &= \{\text{high fever}\},\\ S_2 &= \{\text{skin rash}\}.\end{aligned}$$

Clinical studies report that these symtoms appear with the following frequencies:

$$\begin{aligned}\mathbb{P}\left(S_1 \mid D\right) &= 0.80,\\ \mathbb{P}\left(S_2 \mid D\right) &= 0.75,\\ \mathbb{P}\left(S_1 \mid D^c\right) &= 0.05,\\ \mathbb{P}\left(S_2 \mid D^c\right) &= 0.02.\end{aligned}$$

Assume S_1 and S_2 are *conditionally* independent given D and given D^c.

(a) Compute $\mathbb{P}\left(D \mid S_1\right)$ and $\mathbb{P}\left(D \mid S_2\right)$.

(b) Using the *conditional* independence assumption, compute $\mathbb{P}\left(D \mid S_1 \cap S_2\right)$.

(c) Are the events S_1 and S_2 unconditionally independent? Justify quantitatively.

Exercise 4.12

The table below contains 12 short movie reviews that have been manually labeled as either positive ($y = 1$) or negative ($y = 0$). For every review five binary features are recorded indicating whether the corresponding word appears in the text:

review i	`great`	`boring`	`plot`	`acting`	`slow`	y
1	1	0	1	1	0	1
2	1	0	1	0	0	1
3	0	0	1	1	0	1
4	1	0	0	1	0	1
5	0	1	0	0	1	0
6	0	1	1	0	1	0
7	0	0	0	1	1	0
8	1	0	0	0	1	0
9	0	1	1	1	0	0
10	1	1	1	0	0	0
11	0	1	1	1	0	1
12	0	0	1	0	1	1

Let the feature vector $\vec{X} = (X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)})$ correspond to the presence of the words `great`, `boring`, `plot`, `acting`, `slow`, respectively, and let $Y \in \{0, 1\}$ denote the sentiment class.

(a) Compute the probabilities $\mathbb{P}(\{Y = 1\})$ and $\mathbb{P}(\{Y = 0\})$ from the table.

(b) Consider a new review whose feature vector is

$$\vec{x} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix}^\top$$

Using the naïve Bayes classifier, compute the class likelihoods $\mathbb{P}\left(\{Y = 0\} | \{\vec{X} = \vec{x}\}\right)$ and $\mathbb{P}\left(\{Y = 1\} | \{\vec{X} = \vec{x}\}\right)$. If you need to use Laplace smoothing, clearly indicate why.

(c) Which class does the classifier predict?

Exercise 4.13

Researchers on planet Zog are building a classifier that decides whether a brand new jelly bean is Yummy ($y = 1$) or Yucky ($y = 0$). For each candy they record five quirky binary features:

Feature j	Meaning of $\{X^{(j)} = 1\}$
1	the bean glows in the dark (`glow`)
2	the bean fizzes when bitten (`fizz`)
3	the bean feels slimy (`slimy`)
4	the bean crunches loudly (`crunch`)
5	the bean whistles when shaken (`whistle`)

The training data of 14 beans are listed below.

bean i	glow	fizz	slimy	crunch	whistle	y
1	1	1	0	1	0	1
2	1	0	0	1	0	1
3	0	1	0	0	0	1
4	1	1	0	0	0	1
5	0	0	1	0	1	0
6	0	1	1	1	0	0
7	0	0	1	0	0	0
8	1	0	1	0	1	0
9	0	1	1	0	1	0
10	1	0	1	1	1	0
11	0	1	0	1	0	1
12	1	0	0	0	0	1
13	0	0	1	1	1	0
14	1	1	1	0	1	0

(a) Compute the probabilities $\mathbb{P}(\{Y = 1\})$ and $\mathbb{P}(\{Y = 0\})$ from the table.

(b) A brand-new jelly-bean has feature vector

$$\vec{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}^\top \quad (\texttt{fizz} + \texttt{whistle} \text{ only}).$$

Estimate the probabilities $\mathbb{P}\left(\{Y = 0\} | \{\vec{X} = \vec{x}\}\right)$ and $\mathbb{P}\left(\{Y = 1\} | \{\vec{X} = \vec{x}\}\right)$. If you need to use Laplace smoothing, clearly indicate why.

(c) Which class does the classifier predict?

Exercise 4.14

Consider flipping two fair coins (one *gold* and one *silver*) and spinning a fair three-sided spinner labelled $\{1, 2, 3\}$.

(a) Write the sample space Ω and state its cardinality.

(b) Let

$A = \{\text{the spinner shows an even number}\}, B = \{\text{at least one of the coins shows heads}\}.$

Write the events A and B explicitly as subsets of Ω. Compute $\mathbb{P}(A)$, $\mathbb{P}(B)$, and $\mathbb{P}(A \cup B)$.

(c) Verify the union rule

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$$

using your computations.

Exercise 4.15

Let Ω be a sample space and $A, B, C \subset \Omega$ given events. Prove the **inclusion-exclusion principle**

$$\begin{aligned}\mathbb{P}(A \cup B \cup C) = \mathbb{P}(A) + \mathbb{P}(B) + \mathbb{P}(C) \\ - \mathbb{P}(A \cap B) - \mathbb{P}(B \cap C) - \mathbb{P}(A \cap C) \\ + \mathbb{P}(A \cap B \cap C),\end{aligned}$$

for example, by using the union rule multiple times.

Exercise 4.16

An I.T. access code has the form `LL-DDD`, where each `L` is an uppercase letter excluding `O`, `I`, and `Q`, and each `D` is a digit from 0 to 9. Letters and digits are chosen independently and uniformly at random.

(a) How many distinct access codes are possible?

(b) What is the probability that both letters are consonants?

(c) What is the probability that the three-digit sequence is strictly increasing (for example, `047` or `158`)?

(d) What is the probability that an access code has both properties simultaneously?

Exercise 4.17

A natural history museum has the following distinct taxidermy mounts in its collection:

Birds: B_1, B_2, B_3, B_4, B_5 (where B_1, B_2 are "rare"),
Mammals: M_1, M_2, M_3, M_4,
Reptiles: R_1, R_2, R_3.

They wish to select exactly 7 mounts and arrange them in a row on a shelf, subject to the rules:

- *Type-coverage:* At least one bird, one mammal, and one reptile must appear.
- *Reptile adjacency:* No two reptile mounts are adjacent.
- *Rare-bird condition:* The two rare birds B_1 and B_2 are either *both* included or *both* excluded.

(a) By conditioning on whether B_1, B_2 are in or out, and using inclusion-exclusion or standard "gaps" arguments for the reptiles, derive a closed-form expression (in terms of binomial coefficients and factorials) for the total number of valid arrangements.

(b) Evaluate your expression to obtain the exact numerical count.

(c) Suppose instead the chosen 7 mounts are placed around a circular display (so two arrangements that differ by a rotation are considered the same). How many valid circular arrangements are there?

5 Appendices

A Calculus Background

Theorem A.1 *Local Extrema Occur at Critical Points*

Let f be a function defined on an interval in $\mathbb{R}$, and let c be a point in the domain of f. If f has a local minimum or a local maximum at c, then c is a *critical point* of f. In particular, one of the following must hold:

- $f'(c) = 0$, or
- $f'(c)$ does not exist.

Theorem A.2 *Second Derivative Test*

Let f be a twice-differentiable function on an interval containing a critical point $x = c$, where $f'(c) = 0$. Then:

1. If $f''(c) > 0$, then f has a local minimum at $x = c$.
2. If $f''(c) < 0$, then f has a local maximum at $x = c$.
3. If $f''(c) = 0$, the test is inconclusive, and $x = c$ may be a local minimum, local maximum, or neither.

Theorem A.3 *Second Derivative Test for Functions of Two Variables*

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be twice differentiable and let (x_0, y_0) be a critical point of f, i.e., such that $\nabla f(x_0, y_0) = \vec{0}$. Define the Hessian matrix H at (x_0, y_0) to be given by the 2×2 matrix of second-order partial derivatives as follows:

$$H(x_0, y_0) = \begin{pmatrix} \frac{\partial^2}{\partial x^2} f(x_0, y_0) & \frac{\partial^2}{\partial x \partial y} f(x_0, y_0) \\ \frac{\partial^2}{\partial y \partial x} f(x_0, y_0) & \frac{\partial^2}{\partial y^2} f(x_0, y_0) \end{pmatrix},$$

and let

$$D = \det(H(x_0, y_0)) = \frac{\partial^2 f}{\partial x^2}(x_0, y_0) \frac{\partial^2 f}{\partial y^2}(x_0, y_0) - \left(\frac{\partial^2 f}{\partial x \partial y}(x_0, y_0) \right)^2$$

Then, the following statements hold:

(i) If $\frac{\partial^2}{\partial x^2} f(x_0, y_0) > 0$ and $D > 0$, f has a local minimum at (x_0, y_0);

(ii) If $\frac{\partial^2}{\partial x^2} f(x_0, y_0) < 0$ and $D > 0$, f has a local maximum at (x_0, y_0);

(iii) If $D < 0$, (x_0, y_0) is a saddle point;

(iv) And if $D = 0$, the test is inconclusive.

The following result extends the second derivative test to functions of many variables; we include it here mainly for reference.

Theorem A.4 *Second Derivative Test for Functions of Several Variables*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable scalar-valued function defined on $\mathbb{R}^n$ and let $\vec{x}_0 \in \mathbb{R}^n$ be a critical point where $\nabla f(\vec{x}_0) = \vec{0}$. Assume the Hessian matrix $Hf(\vec{x}_0)$ is invertible. Then:

1. If $Hf(\vec{x}_0)$ is positive definite (equivalently, all eigenvalues are strictly positive), then f attains a strict local minimum at $\vec{x}_0$.
2. If $Hf(\vec{x}_0)$ is negative definite (equivalently, all eigenvalues are negative), then f attains a strict local maximum at $\vec{x}_0$.
3. If $Hf(\vec{x}_0)$ has both positive and negative eigenvalues, then $\vec{x}_0$ is a saddle point of f.
4. In all other cases, the test is inconclusive.

B Linear Algebra Background

Definition B.1 *Dot product*

If $\vec{x}, \vec{y} \in \mathbb{R}^n$ are any vectors, then we define their **dot product** (also called their **inner product**) by the formula

$$\begin{aligned} \vec{x}^\top \vec{y} &= \vec{x}^{(1)}\vec{y}^{(1)} + \vec{x}^{(2)}\vec{y}^{(2)} + \ldots + \vec{x}^{(n)}\vec{y}^{(n)} \\ &= \sum_{s=1}^{n} \vec{x}^{(s)}\vec{y}^{(s)}. \end{aligned}$$

Theorem B.2 *Properties of the vector dot product*

Let $\vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$ be any fixed vectors. Then the dot product satisfies the following properties:

1. Symmetry:
$$\vec{x}^\top \vec{y} = \vec{y}^\top \vec{x}.$$

2. Bilinearity:

$$(a\vec{x} + b\vec{y})^\top \vec{z} = a\,(\vec{x}^\top \vec{z}) + b\,(\vec{y}^\top \vec{z}), \quad \vec{x}^\top (a\vec{y} + b\vec{z}) = a\,(\vec{x}^\top \vec{y}) + b\,(\vec{x}^\top \vec{z}).$$

3. Positive-definiteness:

$$\vec{x}^\top \vec{x} \;\geq\; 0, \text{ and } \vec{x}^\top \vec{x} = 0 \iff \vec{x} = \vec{0}.$$

4. Induced norm:

$$\|\vec{x}\|^2 = \vec{x}^\top \vec{x}.$$

Theorem B.3 *Cauchy-Schwarz Inequality*

Let $\vec{x}, \vec{y} \in \mathbb{R}^n$ be any two vectors. Then the dot product $\vec{x}^\top \vec{y}$ of x and y satisfies

$$\vec{x}^\top \vec{y} \leq \|\vec{x}\|\|\vec{y}\|,$$

with equality if and only if $\vec{x}$ and $\vec{y}$ are linearly dependent.

Theorem B.4 *Rank-Nullity Theorem*

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a fixed $m \times n$ matrix. Then

$$\dim\big(\ker \mathbf{A}\big) \;+\; \dim\big(\operatorname{im} \mathbf{A}\big) \;=\; n.$$

6 Index

p-loss, 12

absolute loss, 12
augmented feature vectors, 46

Binary variables, 5
binomial coefficients, 109

Categorical variables, 5
Combinatorics, 108
complement trick, 106
conditional probability, 113
conditionally independent, 117
constant model, 11
continuous sample spaces, 105
convex function, 75
convex set, 75

data classification, 118
data transformation, 73
degree, 69
design matrix, 46
design vectors, 46
deterministic models, 104
discrete sample spaces, 105
dot product, 133

empirical risk function, 7
entropy loss, 16
epigraph, 75
event, 104

factorial, 109
feature dimension, 18
features, 5
finite difference approximation, 82

general linear model, 52
gradient descent algorithm, 79

hyperparameter, 84

inclusion-exclusion principle, 130
independent, 112
inner product, 133
input variables, 5
interaction design matrix, 74
interaction terms, 73
intercept, 34
intercept (or bias) vector, 52
intercept-free multiple linear regression model, 63

Lagrange multiplier, 92
Laplace smoothing, 122
Lasso regression, 84
law of total probability, 114
learning rate, 80
line of best fit, 34
linear model, 6
loss function, 6

maximum risk, 28
mean squared error (MSE), 40
midrange, 28
model, 5
Model training, 6
multiple linear regression model, 44
mutually independent, 112

naïve Bayes assumption, 120
normal equations, 35
Numerical variables, 5

output variables, 5
overfitting, 72

Parameters, 5
partition, 114
penalty, 84
polynomial feature vector, 69
polynomial regression, 69
probability measure, 104

Regularization, 84
Ridge regression, 84
risk function, 7

sample space, 104
sample space of binary feature vectors, 107
simple linear model, 34
Simple linear regression, 34
simple polynomial regression model, 69
slope, 34
square loss, 6, 11
stationary conditions, 94
strictly convex function, 76
supervised learning, 11

target dimension, 18
testing data, 7
training data, 7
transformed design matrix, 70

Underfitting, 68
union rule, 106
unsupervised learning, 12

vector-valued constant model, 18, 41
vector-valued simplified linear model, 21

zero-frequency problem, 122