1

Introduction

The field of machine learning (ML) has been advancing very rapidly since 2010 when it gained momentum in both academia and industry. In the intervening years, many innovative ideas have been implemented and applied to some very challenging data science problems with great success. Underpinning the work in these areas are the mathematical foundations within the field of probability and statistics. The relationships between the three areas of data science, machine learning, and probability and statistics are conceptualized in Figure 1.1. The **interdependence of the three disciplines** is perhaps the most important point to note here. Data science relies heavily on probability and statistics and the machine learning tools that build the models for a particular application domain. Applications drive the investigation of the type of mathematics and machine learning capabilities that need to be developed. Likewise, machine learning relies on the foundations that lie in probability and statistics and also on the requirements of the applications that will eventually be used to develop the necessary tools. There is a symbiotic relationship between these three disciplines. This book places a strong emphasis on all three areas in equal measure to introduce the ideas and concepts to be described. As such, some familiarity with one of these three areas will be helpful in navigating through the material to be presented.

This book is about robust machine learning. It addresses an under-reported and often overlooked aspect of data science which is the effect of outliers on the model building process. A prediction is only as good as the model on which it is based, and if the model is faulty, so is the prediction. In particular, outliers and anomalies in datasets can lead to inaccurate models, resulting in bad business decisions, questionable explanations of cause-and-effect, incorrect conclusions, or inaccurate medical diagnoses, just to name a few. Even one outlier can render a model unusable if it happens to be in the wrong place. Robust methods are better-suited to data science, especially when outliers are present. ML practitioners have not yet fully embraced a class of robust techniques that would provide more reliable models and more accurate predictions than is possible with present-day methods. The overall goal of this book is to provide the rationale and techniques for robust ML and then build on that material toward robust data science.

The topics to be covered in this book span a wide variety of subjects and there is substantial ground to cover to gain *deep knowledge* in this field. This opening chapter covers introductory material and a number of basic concepts in machine learning. Initially, descriptions of robust machine learning and robust data science are provided as a backdrop for the material to be presented in the rest of the book. Then, we focus on robustifying one ML algorithm. There are several options to choose from in this initial chapter to demonstrate the value of using robust methods. Among the options, the one selected is simple

Fundamentals of Robust Machine Learning: Handling Outliers and Anomalies in Data Science,
First Edition. Resve Saleh, Sohaib Majzoub and A. K. Md. Ehsanes Saleh.

Companion Website: www.wiley.com/go/HandlingOutliersandAnomaliesinDataScience1e

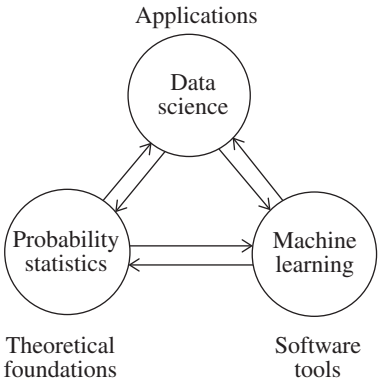


Figure 1.1 Synergistic relationships between data science, machine learning, and probability and statistics.

and yet provides a clear picture of the power of robust methods in general. Since clustering is one of the well-known applications of machine learning, it will be used as the first vehicle to understand the way that robust methods can be applied to handle outliers in datasets. It will also serve as a template for the rest of the book.

1.1 Defining Outliers

Outliers are typically a small subset of data that deviates from the majority of the data within the same dataset. Sometimes the true data is differentiated from outliers by calling them **inliers**. The sources of outliers can vary significantly. Outliers can occur naturally as part of the characteristics of the data being collected. Outliers can also be due to significant noise in the environment that might be unavoidable. More commonly, they are associated with measurement error, or instrumentation error. Another source is due to human error such as typographical errors or perhaps misinterpreting the units of measure of a device. If there are large variances in the features of a dataset and a small amount of data, this could lead to some of the data being declared as outliers. And other times, it is just inherently part of the data collection process. Unusual or extreme outliers are often referred to as **anomalies**.

There are basically two options of handling outliers. The first is to build machine learning tools that are tolerant to outliers. These are the robust machine learning tools. This involves understanding how the current ML tools work and then seeking an alternative approach that is more effective in the presence of outliers. Conceptually, the current approaches in ML seek the mean of the data to build a model. Robust methods seek the median instead which makes the models much more stable and reliable. This is one of the main thrusts of the book. The problems surrounding the use of the mean will be presented, followed by the advantages of using the median. More generally, the first approach is to convert a non-robust machine learning method into a robust one. Many of the existing techniques are not robust and those should be identified and updated accordingly. However, if a method is inherently robust, it should be preferred over its non-robust counterpart.

A second option is to detect and remove outliers and then apply the current ML tools or extract outliers from a dataset for further inspection. In current practice, there is a large investment in non-robust methodologies that are hard to change or replace, so it is easier to find ways to eliminate outliers and use the existing infrastructure. Since robust methods are not widely used in Python (or well-understood), it seems easier to remove outliers rather than build a new infrastructure based on robust methods.

However, as appealing as it may sound, it is not straightforward and requires an extra level of care to ensure that inliers are not inadvertently removed with outliers. There are other cases where the goal of the project is to find the outliers or anomalies in the dataset rather than to build a model. The detected outliers would be inspected to learn something about the anomalies or any unusual aspects of the data being collected. And finally, there are statistical tools and procedures that rely on the data being outlier-free in order to produce meaningful results (e.g. correlation coefficients and analysis of variance). Hence, the secondary thrust of the book is to describe efficient and effective methods to detect and remove both outliers and anomalies.

1.2 Overview of the Book

With the background and preliminaries covered, we will now move to the main thrust of this book which is the study of standard machine learning techniques and their robust counterparts. Table 1.1 provides a roadmap of the topics to be described and their associated chapters.

1.3 What Is Robust Machine Learning?

Machine learning involves the development of software that analyzes data and produces predictive models based on algorithms and mathematics developed by computer scientists, engineers, and statisticians. One can view it as the programming portion of the problem to build software tools and flows to process data in furtherance of some specific data science objective. Most of the existing techniques are essentially based on finding the **mean** of the data. Robust methods to be described in this book are based on finding the **median** of the data. Thus far, robust methods have not seen widespread use due to their somewhat mysterious nature. These techniques borrow heavily from *robust statistics* (see Maronna et al. 2019) which is a well-known and long-standing field in its own right. However, robust methods have only been of recent interest in machine learning. The objective of this chapter, and indeed the whole book, is to demystify robust methods so that more data scientists can gain access to such methods when the need arises.

Truth be told, most realistic datasets have outliers in one form or another. Some are harmless relative to the rest of the data while others may cause significant errors, especially during inference or prediction. The problem is that it is difficult to know *a priori* which case you are dealing with. You may not be able to tell if the results are reliable unless additional steps are taken to ensure reliability. If, for example, the outliers could be removed in some effective manner, then the traditional methods are quite effective. However, outlier detection and removal are prone to error so this may not be a reliable approach either. It is often considered to be tampering with the data. Unless you can successfully diagnose and detect outliers in large datasets, the models generated by standard ML methods on these modified datasets should be viewed as suspect. The more prudent approach is to employ machine learning tools that use robust procedures wherever possible to avoid the problems altogether.

Generally speaking, robust ML is the development of tools, procedures, and methodologies that are tolerant to outliers. In its most simplistic form, it implies the use of the median (or some neighboring quantile) rather than the mean thereby producing outlier-tolerant models. It also includes the detection and removal of outliers using robust ML. Of course, robust methods introduce a set of different issues that must be addressed before they can be used effectively. As we will show throughout this book,

Table 1.1 Topics, chapters, and descriptions.

Topic	Chapter	Description
Robust Clustering	1	k-means clustering algorithm Robust k-medians clustering algorithm
Robust Linear	2	L_1, L_2, Huber loss functions
Regression	3	log-cosh loss function
	4	Outlier detection, metrics (MSE, MAE), robust standardization
Robust Models	5	Penalty functions (ridge, LASSO, aLASSO)
via Regularization	6	Model generalization, model complexity
Quantile Regression	7	Replacing the quantile check function with a flexible log-cosh function
Robust Binary	8	Logistic regression
Classification		Cross-entropy vs. log-cosh loss functions, SVM, kNN, random forest
Neural Networks for Binary Classification and Classification Metrics	9	Activation functions: relu, sigmoid, training, backprop, gradient descent, cross-entropy vs. log-cosh loss functions recall, precision, F_1 score, AUROC
Multi-class Classification	10	Categorical loss, softmax activation
and Optimization		Modified National Institute of Standards and Technology (MNIST) dataset, Adam optimization
Anomaly Detection and Evaluation Metrics	11	kNN, Isolation Forest, density-based spatial clustering of applications with noise (DBSCAN), MADmax, precision, recall, AUROC
Application to Data Science and Artificial Intelligence (AI)	12	Boston housing, Titanic datasets climate change time series (ARIMA) explainable AI (XAI)

the advantages of robust ML greatly outweigh the disadvantages. Our goal is to provide side-by-side comparisons of traditional methods with robust methods so that the reader can make their own judgments about the value of robust ML.

1.3.1 Machine Learning Basics

Before launching into the key ideas of this book, it is useful to provide some background material on ML and robust methods. ML grew out of the computer science field in the 1980s but went dormant for a time in the 1990s, during which it was viewed as a dead-end field. It experienced a reawakening around 2000

and a significant resurgence around 2010. It continues to gain traction and momentum to this day with the advent of generative AI. Of course, in the 1980s, computers were relatively slow, datasets were not readily available, and algorithms for ML were in their infancy. Fast forward a few decades and we find the landscape has changed dramatically. Today, access to high-speed computing is at everyone's fingertips, data is ubiquitous, and many advanced ML algorithms have been developed, implemented, and tested. They are now widely available through open-access online mechanisms.

Some of the algorithms in ML are grounded in statistical theory, while others are based on nonparametric methods or a set of heuristics. We will study methods that fall into all these categories. There are three main varieties of ML tools: supervised learning, unsupervised learning, and reinforcement learning.[1] We focus only on the supervised and unsupervised learning methods in this book. To understand these two methods, consider a dataset given by a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of observations (rows in the dataset) and d is the number of variables (columns in the dataset). The "ground truth" or true responses associated with each observation is optionally provided in a vector $\boldsymbol{y} \in \mathbb{R}^n$. To contrast the two categories, supervised ML requires both $\boldsymbol{X}$ and $\boldsymbol{y}$, while unsupervised ML requires only $\boldsymbol{X}$.

For example, linear regression, logistic regression, neural networks, k-nearest neighbors, tree-based methods, and so on fall into the supervised category. In this book, we focus on the robustness aspects of these methods. On the other hand, data clustering and anomaly detection are examples of unsupervised learning methods. We will discuss robust clustering in this chapter as a way of introducing robust concepts and their value in machine learning. Anomaly detection will be covered in Chapter 11.

For supervised learning, we mainly consider two types of problems: **regression** and **classification**. Figure 1.2 shows the two cases graphically. On the left side, we show a set of points that are fitted with a line, where the x-axis is the *year* and the y-axis is the number of *sales* in that *year*. Fitting a line to the data is called *linear regression*. The ML objective in this case is to use the data to find the **slope and intercept**

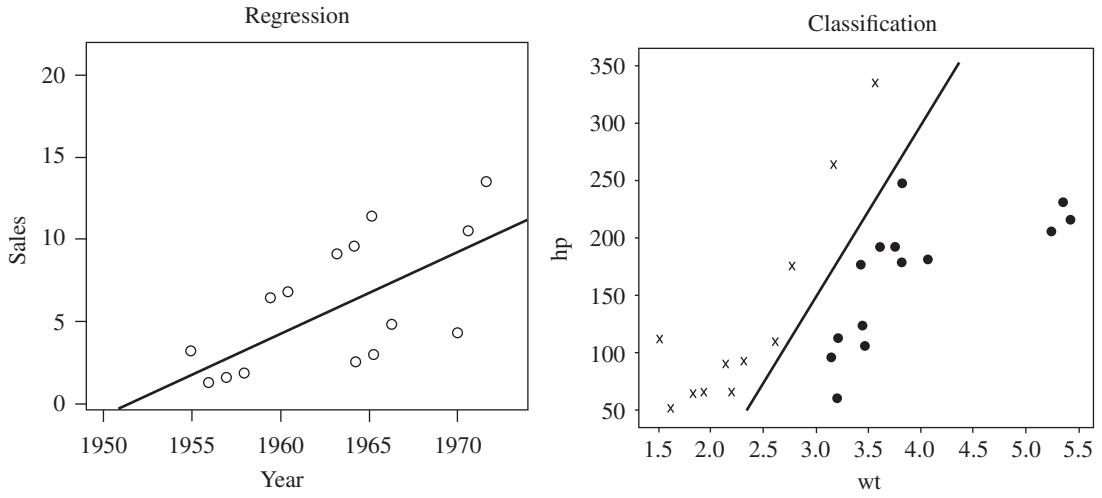


Figure 1.2 Linear regression vs. linear classification.

1 Although an important topic in ML, reinforcement learning is outside the scope of this book.

of the line that "best fits" the given points shown in the plot. The best fit is based on some objective function used during optimization. We are, in effect, training the model to learn about the dataset. Once the estimates of these parameters are obtained, predictions can be made for values that are not in the original dataset. For example, we can now test our model by asking the question: "what is the expected number of *sales* for *year* = 1968?" and the answer comes back as "*sales* = 8.2." We did not have this value in the dataset before but now we are able to make predictions using the fitted line. Building a prediction model is the essence of machine learning.

The figure on the right shows two sets of points, represented using symbols "•" and "x," respectively. Each symbol represents a distinct class. The ML objective here is to establish a line that best separates the "•" points from the "x" points. Finding the dividing line between the two classes is referred to as *linear classification*. The data is linearly separable in this case and the separating line does a good job of defining a *decision boundary* between the two sets of points. We say that the model was trained using the given data, which actually means that we optimized an objective function to produce the slope and intercept parameters. We can now ask another question: "if a new point is introduced at wt = 2.5 and hp = 90, what class does it belong to?" and the answer would come back as "it belongs to the 'x' class." A large portion of the supervised machine learning tools involve this type of classification.

1.3.2 Effect of Outliers

Now consider the effect of outliers on the two cases above. This is shown in Figure 1.3. On the left side, there is one new data point which is an outlier. The effect on traditional linear regression is to shift the line and make it more horizontal than in Figure 1.2. Now if the question is asked, "what is the expected *sales* for *year* = 1968?," the answer comes back as "*sales* = 6.5" which is not accurate. Apparently, the one outlier has produced misleading results. If used to make a business decision, this outlier may prove to be very costly.

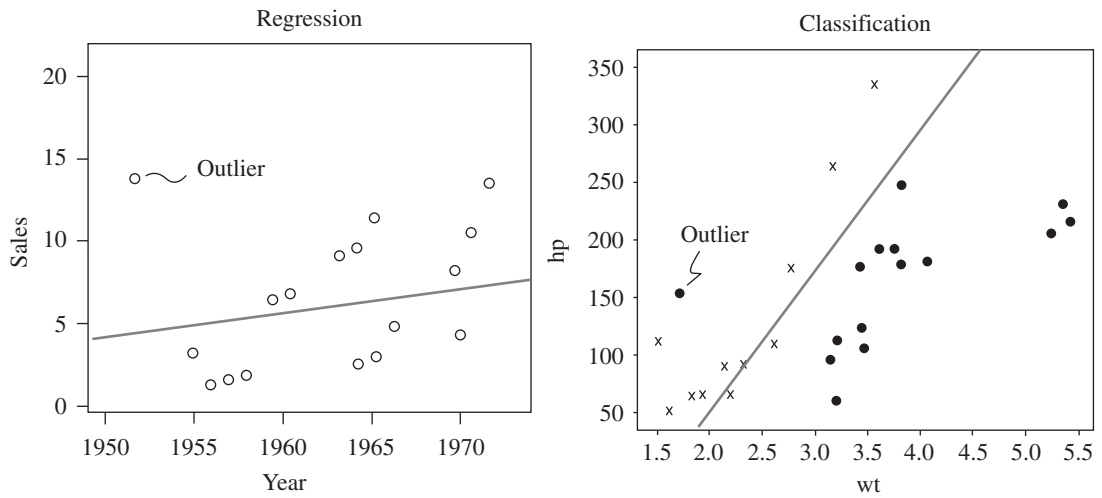


Figure 1.3 Impact of outliers in regression and classification.

In the second case on the right side of Figure 1.3, we place an outlier by adding a "•" on the left side of the boundary, where members of the "x" class reside. The effect of this outlier is to shift the line toward it, relative to the corresponding one in Figure 1.2. The outlier should not affect the boundary, but it does. When we again ask, "if a new point is introduced at wt = 2.5 and hp = 90, what class does it belong to?," the answer would come back as "it belongs to the '•' class." Of course, this is incorrect. It belongs to the "x" class as we saw earlier. The outlier has moved the decision boundary thereby changing the predicted response. As before, any business decision made using these results may be costly.

Ideally, we would like to obtain the results shown in Figure 1.2 given the data of Figure 1.3. Robust methods offer this possibility along with the methodologies to achieve this outcome. Robust ML involves methods that are suitable for datasets with or without outliers. They produce models as if the data were outlier-free. This is the key benefit of using robust methods. You do not need to wonder what the effect of outliers may be in the dataset or where they are located. You simply use this class of methods on all problems without any concerns about outliers. In some applications, you may be interested mainly in detecting and removing the outliers. This issue is also addressed by robust methods, as detailed in this book.

1.3.3 What Is Robust Data Science?

Data science involves a set of application-specific tasks that focus on creating a coherent dataset from a large number of observations, performing thorough exploratory data analysis (EDA), and carefully interpreting results obtained from machine learning tools and statistical analysis. It is a rather expansive field with many layers of depth and a wide variety of applications. We will briefly describe some of these layers and emphasize the key aspects of this discipline. If we step back and look at statistics and machine learning from a higher perspective, we can understand their roles in data science. Figure 1.4 shows the traditional view. We start with the ground truth in the form of a probability distribution (assumed to be the true distribution of a population). By sampling this distribution, we generate a dataset. Once we have a dataset, then traditional statistics/ML seeks to find a model that best approximates the ground truth (i.e. the original distribution). From this model, we can do whatever we want, such as making predictions. Unfortunately, this is an idealized view of the world of data science.

1.3.4 Noise in Datasets

In the real world, there is noise that will add outlier observations to our dataset. Consider a slightly modified version of Figure 1.4 as shown in Figure 1.5. Here, the data generation path is modified to include additive noise. It is the "signal" plus "noise" that combine to produce more realistic datasets.

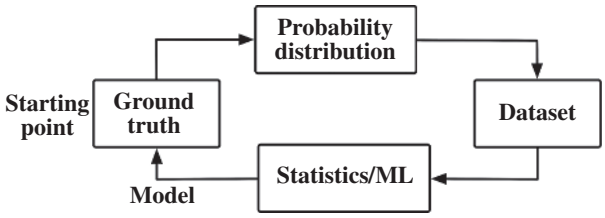


Figure 1.4 Traditional view of the model building process.

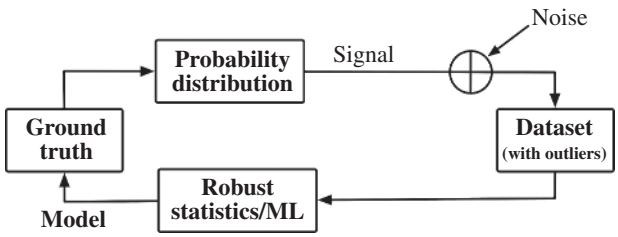


Figure 1.5 Realistic view of the model building process.

This is not the typical Gaussian noise that is usually taken into account. Rather, the noise is disruptive enough to produce bad models if special care is not taken. As a result, in order to construct a model that best approximates the ground truth, we must turn to robust statistics/ML that effectively removes noise and generates a better model. Without robust methods, we may not be able to produce high-quality models that approximate the ground truth when outliers are present.

In robust data science, the focus is on identifying outliers, utilizing data cleaning methods to remove unwanted outliers or applying procedures that are largely insensitive to such outliers. We believe that the field will naturally migrate in this direction, given the fundamentals and benefits of the methods to be presented.

1.3.5 Training and Testing Flows

Data science involves building models using datasets and a number of supervised and unsupervised ML tools. A typical flow for supervised learning is shown in Figure 1.6. We are given an initial dataset with a design matrix $\boldsymbol{X}$ and known responses $\boldsymbol{Y}$. Here, $\boldsymbol{Y}$ is usually a vector but can also be a matrix of responses depending on the application. The dataset is divided into a training set and a test set. We build models using training data and then assess the accuracy using test data. There are many different ways to determine the overall accuracy as described in later chapters. Not shown in the figure are the potential iterations that are needed to experiment with different *hyperparameters*, which are user-specified values that steer the optimization methods toward a desirable model.

The sequence of steps is as follows. Let us assume that the dataset contains outliers. First, the data is randomly split into a training set and a test set. We will have no idea where the outliers have landed; they could be all in the training set, all in the test set, or in both. In practice, we must assume they are in both and develop methodologies and metrics under that assumption. The training data, shown as

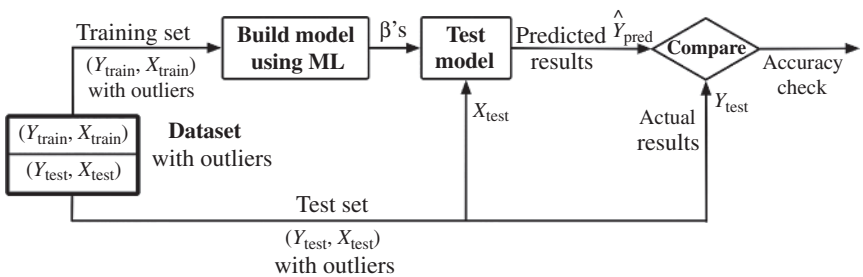


Figure 1.6 Training and test sets for supervised learning.

$(\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$, with potential outliers, is used as input to a selected ML tool, say logistic regression or a neural network, in order to estimate the model parameters, $\boldsymbol{\beta}$. Often, a cross-validation (CV) scheme is used whereby the training set is divided into a number of subsets, called *folds*, so that a more general model can be produced by combining the results of k different folds. Next, the prediction accuracy of the model must be evaluated on an independent test set $(\boldsymbol{X}_{\text{test}}, \boldsymbol{Y}_{\text{test}})$ to properly assess its accuracy. This is carried out by taking $\boldsymbol{X}_{\text{test}}$, possibly with outliers, and applying the trained model to make predictions, and then comparing the predicted results, $\hat{\boldsymbol{Y}}_{\text{pred}}$, against the true values, $\boldsymbol{Y}_{\text{test}}$.

Quality assessment is typically performed using an appropriate set of metrics or statistics.[2] One must be careful when using an error measure with outliers present. Since outliers may appear in the training set, the test set, or both, there may be a need for robust error measures, which will differ for regression and classification. Consider the case where the training set has no outliers but the test set is full of outliers. No amount of training will be able to produce good results on the test set using standard error measures. So we will have to select some suitable robust error measure in order to properly evaluate accuracy.

In unsupervised learning, the dataset is given by $\boldsymbol{X}_{\text{train}}$ but does not involve $\boldsymbol{Y}_{\text{train}}$ or a test set. However, the goal is still to minimize some objective function associated with the specific machine learning task. Whether we are doing supervised or unsupervised learning, the presence of outliers can affect the quality of the model so we have to consider robust ML. With this brief rationale, we can now proceed to the technical details of how robust ML works and where it should be applied.

1.4 Robustness of the Median

Our starting point for robust ML is a brief review of two useful statistics: the mean and the median. The mean is very well-known and forms the basis for the most popular ML methods, whereas the median is less well-known but important for our purposes here.

1.4.1 Mean vs. Median

Given a set of n values, $\boldsymbol{x} = \{x_1, \dots, x_n\}$, the mean is given by

$$\bar{x} = \text{mean}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \tag{1.4.1}$$

which is the sum of the values divided by n.

The median of the set depends on whether n is odd or even. First, we order the values from smallest to largest. After ordering, the new set is denoted by $\{x_{(1)}, \dots, x_{(n)}\}$, where $x_{(1)} \leq \cdots \leq x_{(n)}$. These are referred to as the order statistics of $\boldsymbol{x}$ with $x_{(1)} = \min\{x_i, i = 1, \dots, n\}$ and $x_{(n)} = \max\{x_i, i = 1, \dots, n\}$, and the $x_{(i)}$'s are the ordered observations. If n is odd, we take the middle value in the ordered set,

$$\text{median}(\boldsymbol{x}) = x_{(\frac{n+1}{2})}. \tag{1.4.2}$$

If n is even, we take the average of the two values in the middle of the ordered set. Assuming the above ordering, it can be expressed compactly as follows:

$$\text{median}(\boldsymbol{x}) = \frac{1}{2}\left[x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}\right]. \tag{1.4.3}$$

2 A statistic is any calculation or measurable quantity that can be obtained using data.

Here, the upper value $x_{(\frac{n}{2}+1)}$ and lower value $x_{(\frac{n}{2})}$ are averaged to obtain the median, but technically *any value between the upper and lower values* can serve as the median which implies an infinite number of possible values.

We can already see a problem in that the median seems a little complicated, requires several steps, along with a caveat when n is even, while the mean is easy to understand in one simple equation. This provides insight into why the median has not been embraced as readily as the mean. Furthermore, the mean is always **unique**, whereas the median may or may not be unique. In fact, it can either be unique or any one of an **infinite number of values** in a bounded region. However, the median is more important when it comes to finding the essence of a dataset with outliers, as we will see shortly. It is for good reason that we hear phrases like "the median house price" or "the median household income."

Table 1.2 compares and contrasts these two statistics for some trivial cases. The first set $\{3, 1, 4, 2, 5, 3\}$ produces 3 for both the mean and median. This is consistent with a visual examination of the values in the set. However, if we change one of the values from 3 to 100 to produce the second set, $\{100, 1, 4, 2, 5, 3\}$, the mean increases significantly to 19.2, while the median is 3.5, which is close to the previous value. In this case, the value of 100 would be considered an *outlier* because it is far away from the rest of the data. The mean is greatly affected by one outlier which makes it unstable. On the other hand, the median seems rather stable in the presence of one outlier.

Continuing the exercise, if we examine Set 3 listed as $\{100, 100, 1, 4, 2, 5, 3\}$ in the final column of Table 1.2, we now have two outliers and the mean shifts further beyond the range of the original elements of Set 1 from 3.0 to 30.7. On the other hand, the median maintains relative stability at 4.0. We refer to the median as having the *robustness* property for this reason.

Returning to Set 2 for a moment, if we let $x_{(n)}$ represent the outlier value and set $x_{(n)} = 100$ initially, what happens as $x_{(n)} \to \infty$? The mean tends to ∞ while the median remains stable at 3.5. This is fundamentally why robust methods are needed in ML. If outliers are present, the model cannot be trusted using traditional methods.

1.4.2 Effect on Standard Deviation

The standard deviation is also greatly affected by outliers. Let μ be the population mean, and let $\bar{x}$ of Eq. (1.4.1) be an estimate of μ. Further, let the population variance, σ^2, be defined by

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2,$$

Table 1.2 Comparison of mean and median, and standard deviation (SD), and the normalized median absolute deviation (MADN) on three small datasets.

Measure	Set 1 {3, 1, 4, 2, 5, 3}	Set 2 {100, 1, 4, 2, 5, 3}	Set 3 {100, 100, 1, 4, 2, 5, 3}
Mean	3.0	19.2	30.7
Median	3.0	3.5	4.0
SD	1.41	36.9	47.3
MADN	1.48	2.22	3.0

and the unbiased estimator of σ be given by

$$\text{SD}(\boldsymbol{x}) = \hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{1.4.4}$$

which is the standard deviation (SD) of the sample set. Then, as the outlier increases to infinity, the mean goes to infinity and, as a consequence, the standard deviation goes to infinity. Hence, one outlier can have a domino effect on various other statistics that rely on the mean.

To address this issue, each statistic based on the mean could be replaced by a robust counterpart. A robust statistic for standard deviation is the median absolute deviation (MAD) given by

$$\text{MAD}(\boldsymbol{x}) = \underset{1\le i\le n}{\text{median}}\left(|x_i - \text{median}(\boldsymbol{x})|\right). \tag{1.4.5}$$

A consistent and robust estimator of the population standard deviation based on a **normalized MAD** can be computed as follows:[3]

$$\text{MADN}(\boldsymbol{x}) = k \times \text{MAD}(\boldsymbol{x}), \tag{1.4.6}$$

where $k = 1.4826$. This is used so that $\text{SD}(\boldsymbol{x})$ and $\text{MADN}(\boldsymbol{x})$ are approximately equal for outlier-free data. If we take Set 1 in Table 1.2, the original set with no outliers, then $\text{SD}(\boldsymbol{x}) = 1.41$ and $\text{MADN}(\boldsymbol{x}) = 1.48$. For the case of Set 2 with $x_{(n)} = 100$, $\text{SD}(\boldsymbol{x}) = 39.6$ and $\text{MADN}(\boldsymbol{x}) = 2.22$. Note that $\text{MADN}(\boldsymbol{x})$ is stable compared to $\text{SD}(\boldsymbol{x})$.

If we now change the outlier in Set 2 to $x_{(n)} = 1000$, then $\text{SD}(\boldsymbol{x}) = 407.0$ and $\text{MADN}(\boldsymbol{x}) = 2.22$; that is, MADN remains unchanged. Clearly, a single large outlier can skew basic statistics such as the mean and standard deviation, while the median and $\text{MADN}(\boldsymbol{x})$ remain relatively stable even as $x_{(n)} \to \infty$. For Set 3, with two outliers, $\text{SD}(\boldsymbol{x}) = 47.3$ and $\text{MADN}(\boldsymbol{x}) = 3.0$. We seek this type of stability in most data science problems with outliers.

Exercise: Given the set $\boldsymbol{x} = \{1, 2, 3, 4, 5, 1000\}$, compute the mean, median, $\text{SD}(\boldsymbol{x})$, and $\text{MADN}(\boldsymbol{x})$.

Ans.: mean =1015/6 =169.17, median = 3.5, $\text{SD}(\boldsymbol{x}) = 407$, $\text{MADN}(\boldsymbol{x}) = 2.2$. (Note: to be more precise, the median $\in (3, 4)$. We use the average of 3.5 simply for convenience when we seek a unique value for it). □

1.5 L_1 and L_2 Norms

There are two vector norms that are pertinent to our discussion of traditional methods vs. robust methods: the so-called L_1-norm and L_2-norm. These norms are simply measures of the magnitude of a vector or the distance between two vectors. In particular, the two norms of a vector $\boldsymbol{v}$ with n elements are given as follows:

$$L_1 = \|\boldsymbol{v}\|_1 = \sum_{i=1}^{n}|v_i| = |v_1| + |v_2| + \ldots + |v_n|,$$

3 Here, the word *consistent* means that MADN(x) will produce the true value if the sample size is large enough (see Chapter 2). We will use MADN(x) whenever k is used as a multiply factor of MAD(x). We will use MAD(x) when we do not apply k. Here, k is a factor that depends on the distribution. For a Gaussian normal distribution, $k = 1.4826$.

and

$$L_2 = \|\boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}.$$

The Manhattan distance between two vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$, is given by

$$L_1 = \|\boldsymbol{u} - \boldsymbol{v}\|_1 = \sum_{i=1}^{n} |u_i - v_i|,$$

and the Euclidean distance is given by

$$L_2 = \|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}.$$

Exercise: Let $\boldsymbol{x} = (-1, 2, 3)^\top$. What are the values of the L_1 and L_2 norms?

Ans.: $L_1 = 6, L_2 = \sqrt{14}$. (Note: the $\top$ superscript used in $(-1, 2, 3)^\top$ refers to the transpose of the vector as it is being represented horizontally for convenience rather than vertically.) □

Exercise: Let $\boldsymbol{u} = (-1, 2, 3)^\top$ and $\boldsymbol{v} = (-2, 3, 4)^\top$. What is the distance between these two vectors in terms of L_1 and L_2 norms?

Ans.: $\|\boldsymbol{u} - \boldsymbol{v}\|_1 = 3, \|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sqrt{3}$. □

Often, the phrase "$L2$" may be used even if we square the value of the norm as in

$$L_2^2 = \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = \sum_{i=1}^{n} (u_i - v_i)^2.$$

This looser definition is used in cases where we are comparing relative magnitudes, especially in the case of distance measures, loss functions, or penalty functions. Typically, we are referring to the sum of squares when we use the phrase "$L2$" and sum of absolute values when referring to "$L1$."

Exercise: Let $\boldsymbol{u} = (-1, 2, 3)^\top$ and $\boldsymbol{v} = (-2, 3, 4)^\top$. What are the values of $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$ as compared to the L_2-norm given by $\|\boldsymbol{u} - \boldsymbol{v}\|_2$?

Ans.: $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = 3, \|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sqrt{3}$. One takes the square root of the sum of squares while the other does not take the square root. □

1.6 Review of Gaussian Distribution

One of the most important distributions in statistics is the well-known Gaussian or normal distribution. This subject has been covered in countless papers, books, and online resources. It will be referenced many times throughout this book, so it is presented here for completeness. It is assumed that the reader is well-versed in this area as a prerequisite, but a brief tutorial is provided for review purposes.

The Gaussian distribution has a probability density function (PDF) given by

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{1.6.1}$$

where μ is the mean and σ^2 is the variance. A random variable, X, is said to follow a Gaussian distribution if any sample is distributed according to the equation above. A more compact notation is

$$X \sim \mathcal{N}(\mu, \sigma^2). \tag{1.6.2}$$

This is equivalent to Eq. (1.6.1) except that we are now explicitly stating that X is a random variable distributed as a Gaussian with mean μ and variance σ^2.

The cumulative distribution function (CDF) of the Gaussian is defined as

$$F_X(x) = \mathbb{P}(X \leq x) = \int_{-\infty}^{x} f_X(t)dt. \tag{1.6.3}$$

The CDF gives the probability that a random variable X is less than a specific number x. It is obtained by taking the integral of the PDF from $-\infty$ to x. The total area under any PDF is 1.0 as an axiom of probability theory. Therefore, the results of the integral will always lie in the range $[0, 1]$ as probabilities cannot exceed this range.

We are often interested in computing probabilities from the CDF of the Gaussian distribution but since the integral in Eq. (1.6.3) does not have a closed form, we resort to lookup tables using z-scores which have a normal distribution. The z-score is computed as

$$z = \frac{x - \mu}{\sigma}, \tag{1.6.4}$$

where z is the standardized value of x and $Z \sim \mathcal{N}(0, 1)$. For notational convenience, we refer to the PDF of the Gaussian as $\mathcal{N}(\mu, \sigma^2)$ and the CDF of the Gaussian as $\Phi(x)$.

The multivariate Gaussian distribution for a random variable, $\boldsymbol{X} \in \mathbb{R}^d$, is denoted by

$$\boldsymbol{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1.6.5}$$

where $\boldsymbol{\mu}$ is a $d \times 1$ vector and $\boldsymbol{\Sigma}$ is a $d \times d$ covariance matrix, with the corresponding PDF given by

$$f_X(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{1.6.6}$$

where $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix and $|\boldsymbol{\Sigma}|$ is the determinant.

While the Gaussian distribution is invaluable in statistics, we will later discover that we need to move away from this elegant distribution in favor of other lesser-known distributions to achieve robustness.

1.7 Unsupervised Learning Case Study

We are now ready for our first encounter with a machine learning problem that falls in the category of unsupervised learning. We will take a deep dive into data clustering in the rest of this chapter to introduce and illustrate many of the basic concepts in machine learning with a particular focus on the rationale and proper use of robust techniques. Many of the methods in use today are not robust and therefore our goal is to understand why the existing tools may not be good choices and where robustness can help most.

1.7.1 Clustering Example

Clustering (see James et al. 2023) is a well-known and widely understood problem in ML so it serves as a good entry point into our study of robust methods. The procedure in clustering involves taking a set of unassigned multi-dimensional data points, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, and assigning them to one of K groups or clusters based on their proximity to one another. The value of K is always an integer and serves as a hyperparameter specified by the user based on the application of interest with $K > 1$ (since $K = 1$ is a trivial case). We will walk through the basics of clustering starting with simple problems and then work our way up to the complete set of algorithms for both non-robust and robust methods. In the process, we will follow the loops of Figures 1.4 and 1.5.

Exercise: Let the matrix $\boldsymbol{X}$ be defined as follows:

$$\boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 3 \end{pmatrix}.$$

What is the value of n and d in this case? In this chapter, we will be referring to row i of this matrix using the notation $\boldsymbol{x}^{(i)}$. What is $\boldsymbol{x}^{(2)}$? What is $\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}$?

Ans.: Throughout this book, the notation $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ will be used to refer to a real matrix $\boldsymbol{X}$ with n rows and d columns. In this trivial example, $n = 3$ since there are 3 rows, and $d = 2$ since there are 2 columns. Then, $\boldsymbol{x}^{(2)} = (1, 4)^\top$. Also, $\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} = (0, 2)^\top$. □

1.7.2 Clustering Problem Specification

Given a set of n points in $\mathbb{R}^d$, and an integer hyperparameter, K, the goal of clustering is to assign each point to one of K clusters, $C_1, \ldots, C_K$, based on their proximity to a set of defined cluster centers, $\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_K$, such that it minimizes a distance-based cost function. The cluster centers are computed using the set of points assigned to each cluster. The algorithm used for clustering starts with an initial assignment of points to clusters and then proceeds in an iterative fashion until convergence of cluster assignments is achieved.

To understand the clustering process, consider Figure 1.7. The figure on the left is the original dataset in $\mathbb{R}^2$ represented by variables x_1 and x_2 for each point. The number of clusters is chosen to be $K = 3$ in this case and the clustering procedure after several iterations produces the figure on the right. The representative centers are indicated with the symbol ⊗. The three circular/elliptical boundaries around the data are shown only as a visual aid to identify the different clusters. Often, different colors or symbols are used to represent different clusters.

The problem of clustering is harder than it looks; in fact, it belongs to the class of NP-hard problems, meaning that it cannot be solved in polynomial time as the problem size increases. Therefore, locally optimal solutions are usually sought using "greedy" algorithms, starting with an initial guess for the centers and then repeating the process of clustering the data and recomputing the centers until convergence. Results may vary significantly depending on the choice of initial starting points.

A generic unsupervised clustering algorithm is shown in Algorithm 1.1. It begins by randomly picking K centers. Then, the iterative loop is entered. In the first phase of the loop, K groups are defined by

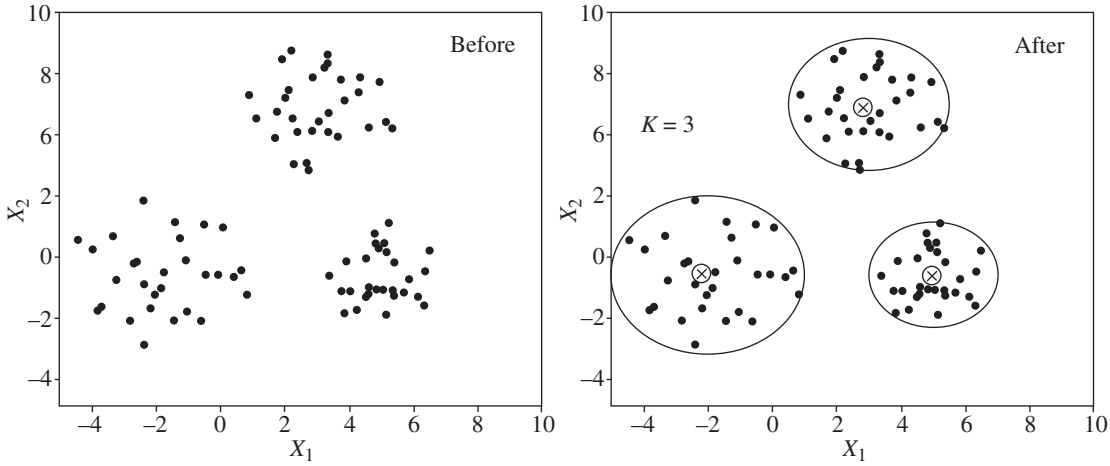


Figure 1.7 Dataset before and after clustering.

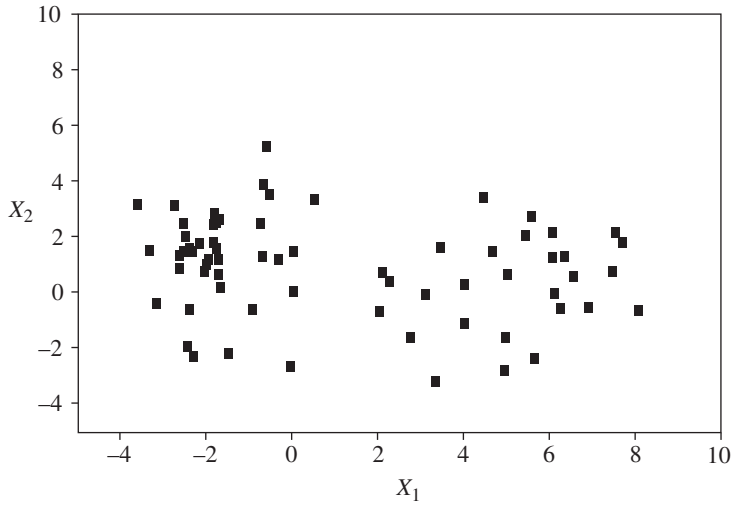
assigning data points to their closest center. In the second phase, the centers of each group are recomputed using the data points assigned to it. Then a suitable cost function is computed for the new arrangement. The iterations continue through the two phases until convergence is achieved. This occurs when the cost function is minimized and the K groups no longer change. The loop is referred to as a Lloyd-style iteration process.

The algorithm is sensitive to the initial guesses so it is worth discussing some of the options for initialization. The simplest approach is to pick K random points from the n total points in the dataset as suggested in the algorithm. Unfortunately, these random values may be close to one another and that is not desirable because it may lead to unbalanced clustering. Another way is to choose $K + m$ initial centers and later remove m centers to produce K clusters. This is sometimes used to find outliers in the data. A third initialization scheme involves picking K centers that are far away from each other. This allows a more balanced set of clusters to form. In any case, some clever method should be used to pick the K centers. Otherwise, the results may not be satisfactory for their intended purpose.

Algorithm 1.1 Generic clustering algorithm using Lloyd-style iteration.

1: **Input:** dataset $\boldsymbol{X}$, number of clusters = K;
2: Initialize centers using K random points of $\boldsymbol{X}$, $\text{cost}_{\text{old}} = 0$, $\text{cost}_{\text{new}} = \text{maxcost}$;
3: **repeat until** $(|\text{cost}_{\text{new}} - \text{cost}_{\text{old}}| < \varepsilon)$:
4: Define K new clusters, $C_1, \ldots, C_K$, by assigning each point to the nearest center;
5: Recompute K cluster centers of $C_1, \ldots, C_K$;
6: $\text{cost}_{\text{old}} = cost_{\text{new}}$;
7: Compute cost_{new} using a distance-based cost function;
8: **end repeat**
9: **Output:** final clusters, $C_1, \ldots, C_K$;

Exercise: Consider the scatter plot of data points below. What is the value of d and a suitable value of K in this case?



Ans.: Here, $d = 2$ because it is a two-dimensional problem, and it appears that $K = 2$ is a suitable choice since there are apparently two clusters. □

1.8 Creating Synthetic Data for Clustering

In the development of ML tools, we will often create representative datasets to validate the workings of the program. These are referred to as *synthetic datasets*. Synthetic data is also used to test corner cases, introduce artificial outliers, and augment existing real-world datasets to improve the quality of the models generated by ML tools. Before we solve the clustering problem, we need a way to generate synthetic datasets that are composed of any number of clusters in a multi-dimensional space. We start with a probability distribution of some sort as depicted in Figure 1.4. For clustering, an appropriate distribution is a linear combination of several Gaussian distributions called a **Gaussian mixture**. For simplicity, we start the case of two clusters and then generalize to K clusters. For testing purposes, we also need to generate clusters of points in 1-dimension, 2-dimensions, or any size up to d-dimensions.

1.8.1 One-Dimensional Datasets

A one-dimensional problem with two Gaussian distributions in a mixture takes the following form:

$$f_X(x) = \pi_1 \mathcal{N}(\mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(\mu_2, \sigma_2^2), \tag{1.8.1}$$

where π_i are the probability weighting factors (**not** the numerical value π) associated with each distribution of the form $\mathcal{N}(\mu_i, \sigma_i^2)$. By definition, $\pi_1 + \pi_2 = 1$. If we had K Gaussian distributions in the mixture,

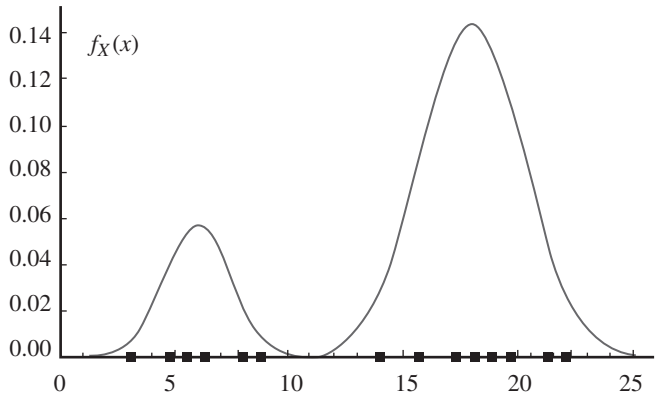


Figure 1.8 One-dimensional Gaussian mixture.

then we would require that

$$\sum_{i=1}^{K} \pi_i = 1$$

in order to satisfy a basic probability axiom associated with a valid PDF.

Example: The graph shown in Figure 1.8 represents a one-dimensional Gaussian mixture. It is based on the Gaussian mixture model of Eq. (1.8.1). The parameters of the mixture are $\pi_1 = 0.2$, $\pi_2 = 0.8$, $\mu_1 = 6$, $\sigma_1^2 = 2$, $\mu_2 = 18$, and $\sigma_2^2 = 5$. When we draw a sample of 14 observations from such a distribution, we obtain a set of random points as depicted along the x-axis. This comprises the dataset that we are given without any prior knowledge about the original distribution. The objective is to identify the two clusters using this data and reconstruct the original distribution on which the sample was based, as represented earlier in Figure 1.4. That is, we want to build a model of an unknown distribution from which the data arose given only the random sample of points. The key concept is that data is always tied to some unknown distribution. The best we can do with the given data is to build an approximate model of it using machine learning. □

Exercise: What is the area under the curve (AUC) of the Gaussian mixture of Figure 1.8 from $-\infty$ to ∞?

Ans.: The AUC = 1.0 by definition of a valid probability distribution. In fact, all valid PDFs have normalizing constants to ensure that they integrate to 1. □

1.8.2 Multidimensional Datasets

Consider a more realistic d-dimensional Gaussian distribution denoted by $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)^\top$ and $\boldsymbol{\Sigma}$ is a $d \times d$ nonsingular covariance matrix. In its most general form, we have the following Gaussian mixture:

$$f_X(\boldsymbol{x}) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}_d(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i). \tag{1.8.2}$$

Thus, we may define mixtures of two d-dimensional Gaussian distributions as

$$f_X(\boldsymbol{x}) = \pi_1 \mathcal{N}_d(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \pi_2 \mathcal{N}_d(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2). \tag{1.8.3}$$

In the rest of this chapter, we will assume that the covariance matrix takes the simplified form of $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_d$, where $\boldsymbol{I}_d$ is a $d \times d$ identity matrix. This is suitable for circular or spherically shaped distributions of data. Then, for $K = 2$ and an arbitrary d,

$$f_X(\boldsymbol{x}) = \pi_1 \mathcal{N}_d(\boldsymbol{\mu}_1, \sigma_1^2 \boldsymbol{I}_d) + \pi_2 \mathcal{N}_d(\boldsymbol{\mu}_2, \sigma_2^2 \boldsymbol{I}_d). \tag{1.8.4}$$

In the case when $K = 2$ and $d = 2$, we have

$$f_X(\boldsymbol{x}) = \pi_1 \mathcal{N}_2(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \pi_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2). \tag{1.8.5}$$

where $\pi_1 + \pi_2 = 1$, $\boldsymbol{\mu}_1 = (\mu_{11}, \mu_{12})^\top$, $\boldsymbol{\mu}_2 = (\mu_{21}, \mu_{22})^\top$,

$$\boldsymbol{\Sigma}_1 = \sigma_1^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{\Sigma}_2 = \sigma_2^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

For notational convenience, we can define a set $\Theta = \{\pi_1, \pi_2, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \sigma_1, \sigma_2\}$ that contains all the true parameters. Data can be generated from the probability distribution of this two-dimensional Gaussian mixture in the same manner as described for the one-dimensional case. Based on the data, we can estimate the parameters of the mixture to reconstruct the distribution. When we compute these parameters, we will use $\hat{\Theta} = \{\hat{\pi}_1, \hat{\pi}_2, \hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \hat{\sigma}_1, \hat{\sigma}_2\}$, where the "hat" represents an estimate of each parameter. This would constitute the **model** of the distribution.

Example: Consider the data in Figure 1.9. It was produced by sampling an unknown Gaussian mixture. We want to express it in the form given in Eq. (1.8.5). How can we do this? Assume we can identify the two clusters and the points belonging to each one using some clustering algorithm. Then we find that the centers of the two clusters are (1.1,1.2) and (6.9,7.1), respectively, and assuming the covariance matrices of the two clusters are symmetric, we can compute the variances as 1.3 and 2.1, respectively. Also, we find that there are twice as many points in one cluster compared to the other. To write a Gaussian mixture model using Eq. (1.8.5), we set $\hat{\pi}_1 = 1/3$ and $\hat{\pi}_2 = 2/3$. The centers of the clusters are

$$\hat{\boldsymbol{\mu}}_1 = (1.1, 1.2)^\top, \quad \text{and} \quad \hat{\boldsymbol{\mu}}_2 = (6.9, 7.1)^\top.$$

Then the covariance matrices are

$$\hat{\boldsymbol{\Sigma}}_1 = 1.3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \hat{\boldsymbol{\Sigma}}_2 = 2.1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

It turns out that the true values are $\pi_1 = 0.3$, $\pi_2 = 0.7$, $\boldsymbol{\mu}_1 = (1, 1)^\top$, $\boldsymbol{\mu}_2 = (7, 7)^\top$, $\sigma_1^2 = 1$, and $\sigma_2^2 = 2$. The model is quite accurate in this case and it completes the flow shown in Figure 1.4. The only missing step is how to create the clusters in the first place, which is the subject of the next section. □

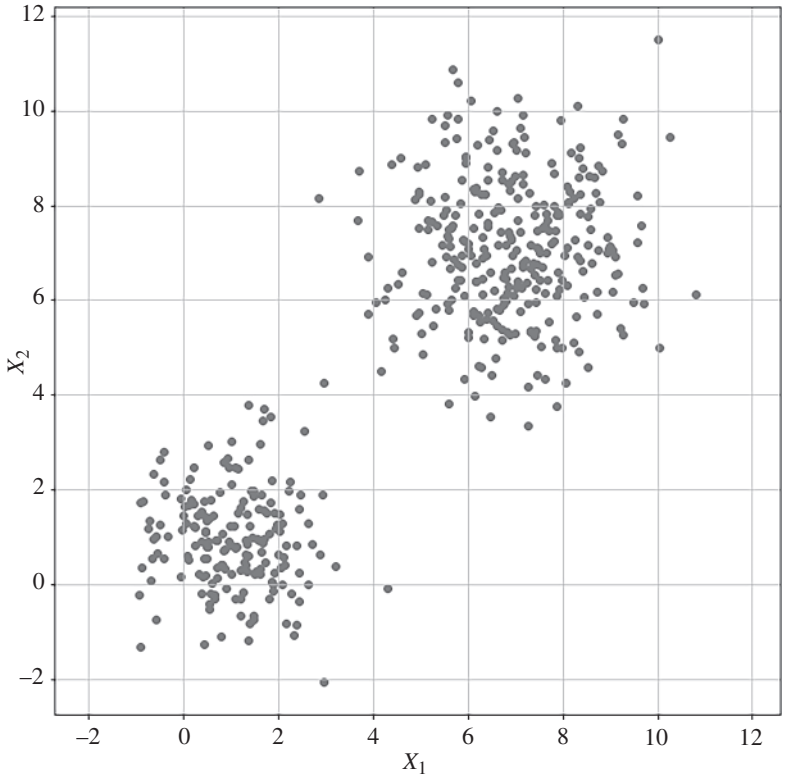


Figure 1.9 Two-dimensional Gaussian mixture.

1.9 Clustering Algorithms

We have already described a Gaussian mixture method of generating data and a generic Lloyd-style iteration loop to create clusters. We now examine two ways to implement Algorithm 1.1. Once we choose a distance measure, a cost function and a centering criteria, the generic algorithm becomes a specific algorithm.

1.9.1 *k*-Means Clustering

The first approach is called *k-means clustering*. In phase 1, points are assigned to their closest center based on an L_2 distance measure. Each distance is computed as follows:

$$\begin{aligned}\text{distance}(i,j) &= \left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right\|_2^2 \\ &= (x_1^{(i)} - \mu_1^{(j)})^2 + (x_2^{(i)} - \mu_2^{(j)})^2 \ldots (x_d^{(i)} - \mu_d^{(j)})^2,\end{aligned}$$

where $\boldsymbol{x}^{(i)}$ is the ith observation of $\boldsymbol{X}$ and $\boldsymbol{\mu}^{(j)}$ is the jth center out of K possible centers. Then each of the $\boldsymbol{x}^{(i)}$, $i = 1, \dots, n$, is assigned to the closest $\boldsymbol{\mu}^{(j)}$, $j = 1, \dots, K$ based on the distance formula. In the second stage, the *centroids* of the clusters are computed using the points assigned to each cluster for all K clusters. This is simply the column-wise mean of the data points assigned to each cluster,

$$\boldsymbol{\mu}^{(j)} = \text{mean}\,(\boldsymbol{C}_j) \;\; \forall j = 1, \dots, K.$$

In the above, $\boldsymbol{C}_j$ is the cluster matrix for cluster j.

The cost is computed as the total sum of all distances between data points inside each cluster to their respective cluster centers for all clusters:

$$\text{cost} = \sum_{j=1}^{K} \sum_{i \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\|_2^2. \tag{1.9.1}$$

In the above, C_j is the set of row numbers of $\boldsymbol{X}$.

The old and new costs are compared relative to a small value, ε, to determine convergence as follows:

$$|\text{cost}_{\text{new}} - \text{cost}_{\text{old}}| < \varepsilon. \tag{1.9.2}$$

The two-phase loop continues until the convergence criterion is satisfied and none of the centers change in two consecutive loops.

Exercise: Let $\boldsymbol{x}^{(i)}$ be the ith observation of matrix $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ given by

$$\boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ \vdots & \vdots \\ 2 & 3 \end{pmatrix}.$$

Only the 1st, 2nd, and 60th rows of the matrix are shown above. Let cluster C_1 consist of only those three points, specifically $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$, and $\boldsymbol{x}^{(60)}$. By our notation, $C_1 = \{1, 2, 60\}$. What is the cluster centroid, which we denote by $\boldsymbol{\mu}^{(1)}$?

Ans.: The data points in C_1 are $\boldsymbol{x}^{(1)} = (1, 2)^\top$, $\boldsymbol{x}^{(2)} = (1, 4)^\top$, and $\boldsymbol{x}^{(60)} = (2, 3)^\top$. The centroid is the coordinate-wise average. Therefore, the centroid is $\boldsymbol{\mu}^{(1)} = ((1 + 1 + 2)/3, (2 + 4 + 3)/3)^\top = (4/3, 3)^\top$. □

Exercise: Let $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ be defined as above. What is the L_2 distance from $\boldsymbol{x}^{(2)}$ to a cluster center $\boldsymbol{\mu}^{(1)} = (4/3, 3)^\top$?

Ans.: The distance is given by

$$\text{distance}(2, 1) = \|\boldsymbol{x}^{(2)} - \boldsymbol{\mu}^{(1)}\|_2^2 = (x_1^{(2)} - \mu_1^{(1)})^2 + (x_2^{(2)} - \mu_2^{(1)})^2.$$

Therefore, $(1 - 4/3)^2 + (4 - 3)^2 = 10/9$. □

Exercise: Let cluster C_1 consist of three elements of $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ given by $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$ and $\boldsymbol{x}^{(60)}$ for the matrix given above. What is the L_2 cost of this cluster assuming $\boldsymbol{\mu}^{(1)} = (4/3, 3)^\top$?

Ans.: The L_2 cost is given by

$$\text{cost}(C_1) = \|\boldsymbol{x}^{(1)} - \boldsymbol{\mu}^{(1)}\|_2^2 + \|\boldsymbol{x}^{(2)} - \boldsymbol{\mu}^{(1)}\|_2^2 + \|\boldsymbol{x}^{(60)} - \boldsymbol{\mu}^{(1)}\|_2^2.$$

Therefore, $[(1 - 4/3)^2 + (2 - 3)^2] + [(1 - 4/3)^2 + (4 - 3)^2] + [(2 - 4/3)^2 + (3 - 3)^2] = 24/9$. □

1.9.2 *k*-Medians Clustering

One problem with *k*-means clustering is that it is not robust to outliers. This issue will be highlighted in sections to follow, but there are many clustering problems that contain outliers and they will disrupt the ability to obtain acceptable results using *k*-means. A logical alternative is to use *k-medians clustering* which employs different distance, cost, and center calculations. The distance is typically given by the Manhattan distance (although Euclidean distance can also be used) as follows:

$$\begin{aligned}\text{distance}(i,j) &= \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\|_1 \\ &= |x_1^{(i)} - \mu_1^{(j)}| + |x_2^{(i)} - \mu_2^{(j)}| + \ldots + |x_d^{(i)} - \mu_d^{(j)}|. \end{aligned} \tag{1.9.3}$$

This L_1 distance measure is used to determine the assignment of observations to clusters. Although this is typically used in *k*-medians, the L_2 distance is also suitable for this purpose. The reason is that distance is a relative measure here so either of the two options may be used. What is crucial for robustness is the computation of the centers. After assignment, the centers are subsequently determined as the column-wise medians of the observations in each cluster:

$$\boldsymbol{\mu}^{(j)} = \text{median}(\boldsymbol{C}_j) \;\; \forall j = 1, \ldots, K,$$

where matrix $\boldsymbol{C}_j$ contains the subset of rows of $\boldsymbol{X}$ that belong in cluster j.

The cost is computed as follows:

$$\text{cost} = \sum_{j=1}^{K} \sum_{i \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\|_1. \tag{1.9.4}$$

The use of the median for center computation is what robustifies the algorithm. The centers are no longer controlled by outliers and reasonable values of centers that truly represent each cluster will be found using this approach. To summarize, both the cost and distance measures of *k*-means can be used in the *k*-medians approach, so long as the centers are computed using the median.

Exercise: Let cluster C_1 consist of 3 elements of $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ given by $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$, and $\boldsymbol{x}^{(60)}$ of the matrix

$$\boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ \vdots & \vdots \\ 100 & 100 \end{pmatrix}.$$

Hence, $C_1 = \{1, 2, 60\}$ in this case. What is the cluster median?

Ans.: The data points in C_1 are $\boldsymbol{x}^{(1)} = (1,2)^\top$, $\boldsymbol{x}^{(2)} = (1,4)^\top$ and $\boldsymbol{x}^{(60)} = (100,100)^\top$. The center is the coordinate-wise median. Therefore, $\boldsymbol{\mu}^{(1)} = (1,4)^\top$. This is about the same as the centroid $(4/3, 3)^\top$ found in an earlier exercise even though this dataset has an outlier at $(100,100)^\top$ which is part of cluster C_1. □

Exercise: Let $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ be defined as above. What is the Manhattan distance from $\boldsymbol{x}^{(2)}$ to a cluster median $\boldsymbol{\mu}^{(1)} = (1,4)^\top$?

Ans.: The Manhattan distance is given by

$$\text{distance}(2,1) = \|\boldsymbol{x}^{(2)} - \boldsymbol{\mu}^{(1)}\|_1 = |x_1^{(2)} - \mu_1^{(1)}| + |x_2^{(2)} - \mu_2^{(1)}|.$$

Therefore, $|1-1| + |4-4| = 0$. □

Exercise: Let cluster C_1 consist of three elements of $\boldsymbol{X} \in \mathbb{R}^{60\times 2}$ given by $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$, and $\boldsymbol{x}^{(60)}$ of the matrix given above. What is the L_1 cost of this cluster assuming $\boldsymbol{\mu}^{(1)} = (1, 4)^\top$?

Ans.: The L_1 cost is given by

$$\text{cost} = \|\boldsymbol{x}^{(1)} - \boldsymbol{\mu}^{(1)}\|_1 + \|\boldsymbol{x}^{(2)} - \boldsymbol{\mu}^{(1)}\|_1 + \|\boldsymbol{x}^{(50)} - \boldsymbol{\mu}^{(1)}\|_1.$$

Therefore, $[|1-1| + |2-4|] + [|1-1| + |4-4|] + [|100-1| + |100-4|] = 2 + 0 + [99 + 96] = 197$. □

1.10 Importance of Robust Clustering

With the data generation and clustering algorithms in place, it is time to compare the two approaches on synthetic datasets. Initially, the dataset will not have any outliers to assess how each method performs on outlier-free datasets. Then, outliers will be inserted to examine the consequences of their presence on each method.

1.10.1 Clustering with No Outliers

Let us return to the simple flow shown in Figure 1.4 and follow a complete sequence around the loop. Consider a Gaussian mixture in $\mathbb{R}^2$ given by

$$f_\Theta(x) = 0.6\,\mathcal{N}_2((-2, 1)^\top, 3\boldsymbol{I}_2) + 0.4\,\mathcal{N}_2((5, 0)^\top, 4\boldsymbol{I}_2), \tag{1.10.1}$$

which is the ground truth in this example. We draw a sample from this distribution to create the dataset, as prescribed by the upper portion of the loop in Figure 1.4. The resulting dataset composed of a total of 60 points is shown in Figure 1.10 and it does not have any outliers. Although not immediately apparent, there are two clusters in the scatter plot since the Gaussian mixture has two terms.

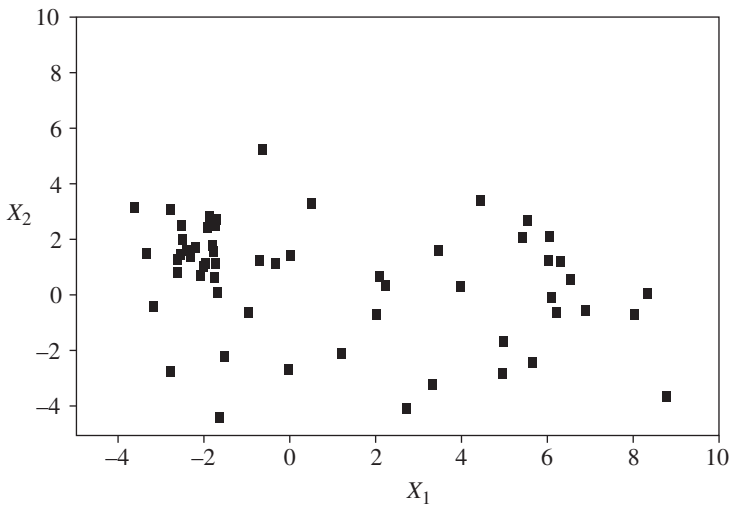


Figure 1.10 Dataset before clustering.

In practice, we are given the data but no access to the ground truth. Assume all we know is that the hyperparameter is fixed at $K = 2$. The question is, can we reconstruct the Gaussian mixture using only the given data to close the loop in Figure 1.4? The answer is yes. We can do this using a clustering algorithm followed by the estimation of the parameters of Θ.

The k-means clustering technique can be invoked using `KMeans` in the `sklearn` Python library. The k-medians approach is not available in `sklearn` so a simple version can be written in Python for comparison purposes.[4]

If we run the `KMeans` algorithm to convergence, the resulting cluster centers are

$$\hat{\boldsymbol{\mu}}^{(1)} = (-1.92, 1.17)^\top \quad \text{and} \quad \hat{\boldsymbol{\mu}}^{(2)} = (4.93, -0.17)^\top.$$

These values are close to the cluster centers associated with the original distribution of Eq. (1.10.1). If we run the k-medians algorithm, we obtain

$$\hat{\boldsymbol{\mu}}^{(1)} = (-1.89, 1.46)^\top \quad \text{and} \quad \hat{\boldsymbol{\mu}}^{(2)} = (5.48, 0.30)^\top.$$

These centers are also close to the original distribution values.

The results are shown graphically in Figure 1.11. The k-means and k-medians methods produce roughly the same two clusters and centers (indicated by $\otimes$ in the figure). Both sets of results are acceptable as they are close to the two centers of the original distribution. To close the loop of Figure 1.4, we need the variances of each cluster by computing the covariance matrix of the form $\sigma^2 \boldsymbol{I}_2$, and also the probability weights based on the ratio of points in each cluster relative to the total number of points. In doing so, using k-means, we obtain:

$$f_{\hat{\Theta}}(x) = 0.6\mathcal{N}_2((-1.9, 1.0)^\top, 2.4\boldsymbol{I}_2) + 0.4\mathcal{N}_2((4.9, -0.2)^\top, 3.8\boldsymbol{I}_2) \tag{1.10.2}$$

and with k-medians, we obtain:

$$f_{\hat{\Theta}}(x) = 0.62\mathcal{N}_2((-1.9, 1.5)^\top, 2.5\boldsymbol{I}_2) + 0.38\mathcal{N}_2((5.5, 0.3)^\top, 3.7\boldsymbol{I}_2) \tag{1.10.3}$$

Both models are reasonably accurate. In practice, we will not be able to compare our model to the ground truth. This is as far as we can go with the given data but we have no way to determine if the results are acceptable. It will be hard to prove our model is correct. But since this data is outlier-free and we know $K = 2$, we can have a high degree of confidence in the correctness of the model in this case.

1.10.2 Clustering with Outliers

So far, we have found k-means and k-medians to be about the same for the given outlier-free dataset. What if one of the points was inadvertently entered manually as $(-400, -400)$ instead of the correct value of $(-4.00, -4.00)$? This outlier along with the original data are shown in Figure 1.12. The outlier is due to human error and we want to investigate the effect on the two competing clustering methods. Note that the outlier would be easy to detect and remove in a two-dimensional setting, but in a higher-dimensional setting, where visualization is not possible, it would be more difficult.

The initial question is: how well do the two methods perform in reproducing the original distribution from which the data arose before noise is added. Referring back to Figure 1.5, the noise is the outlier

4 The k-medians code in Python is available in one of the projects at the end of this chapter.

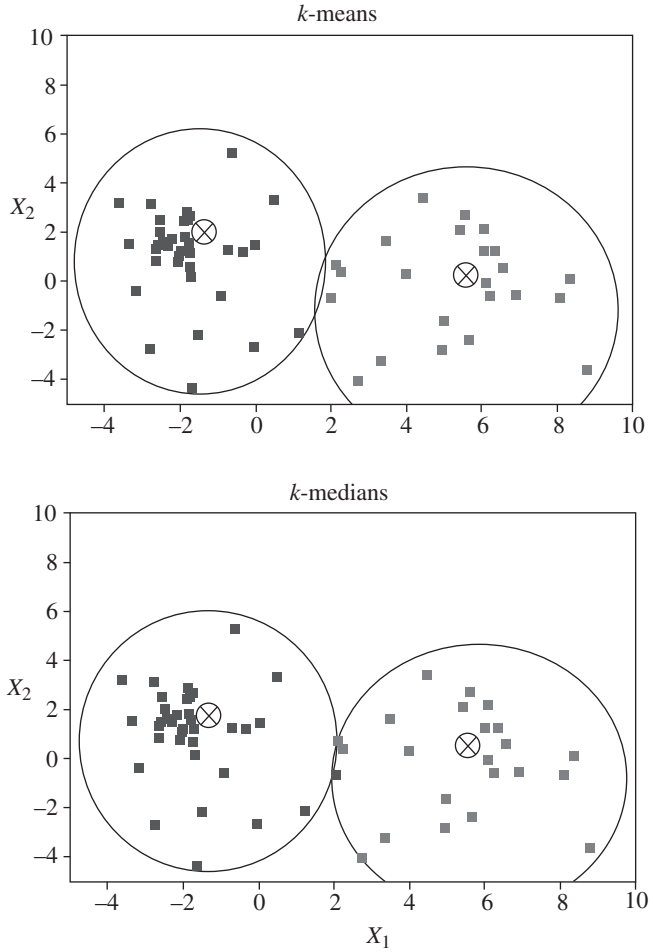


Figure 1.11 *k*-means and *k*-medians clustering.

introduced due to a typographical error. The results of *k*-means are shown graphically in the upper panel of Figure 1.13. One large cluster is created with all the data and another cluster is created containing only the outlier. The outlier cluster is not shown in Figure 1.13 at the scale of the plot but it is clear that one of the cluster centers is forced outside of the region of the data by the *k*-means algorithm because only one is visible in the diagram. The two centers are

$$\hat{\boldsymbol{\mu}}^{(1)} = (-400, -400)^\top \quad \text{and} \quad \hat{\boldsymbol{\mu}}^{(2)} = (1.10, 0.6)^\top.$$

The first cluster has a center that is well outside the cluster region located at $(-400, -400)$. The other center is located in the middle of the true data. It is evident that even a single outlier in the dataset can significantly impact *k*-means clustering. We saw this effect in an earlier section regarding the mean vs. the median and we see it here again in the clustering problem.

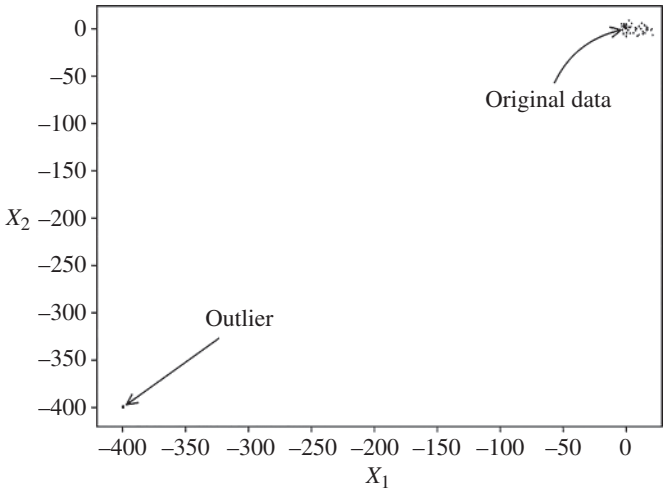


Figure 1.12 Data with 1 outlier at (−400, −400).

Running the k-medians algorithm produces the following centers:

$$\hat{\boldsymbol{\mu}}^{(1)} = (-1.89, 1.33)^\top \quad \hat{\boldsymbol{\mu}}^{(2)} = (5.48, 0.30)^\top.$$

Unlike the k-means case, the centers have not changed very much. This is highly desirable because it indicates that the outlier has not affected the outcome of clustering. This is shown graphically in lower panel of Figure 1.13 where the same two clusters are produced. The k-medians results are relatively stable compared to k-means. This is a key point to emphasize: using a robust method produces superior results in the presence of outliers and similar results with no outliers.

1.10.3 Detection and Removal of Outliers

One way to produce better results using k-means is to detect and remove outliers. This requires some techniques that are themselves robust in nature since the non-robust methods fail to address the issue. Consider the dataset with 1 outlier as described in the previous section. We must first run k-medians clustering since the results from k-means results are not meaningful. More generally, in order to find outliers, a robust method is always part of the methodology.

Once we have the results of robust clustering, we can apply simple rules to remove outliers. One approach is to use the "3-sigma edit" rule. This rule is used to detect and delete any points that are greater than $3\sigma_j$ away from the center of cluster C_j, where $j = 1, \ldots, K$. This rule uses the criteria that, if point $\boldsymbol{x}^{(i)}$ is in matrix $\boldsymbol{C}_j$,

$$\text{delete } \boldsymbol{x}^{(i)} \quad \text{if } \|\boldsymbol{x}^{(i)} - \hat{\mu}^{(j)}\|_2 > 3\hat{\sigma}_j \quad \text{for } i = 1, \ldots, n,$$

where $\hat{\mu}^{(j)}$ and $\hat{\sigma}_j$ are the mean and standard deviation of the points in cluster j. Unfortunately, the computation of the cluster mean and standard deviation will be skewed by outliers so they cannot be trusted.

A better approach is to use a robust form of the 3-sigma edit rule. Since the mean and variance cannot be computed reliably under outlier conditions, robust statistics can be used in their place. Data will only

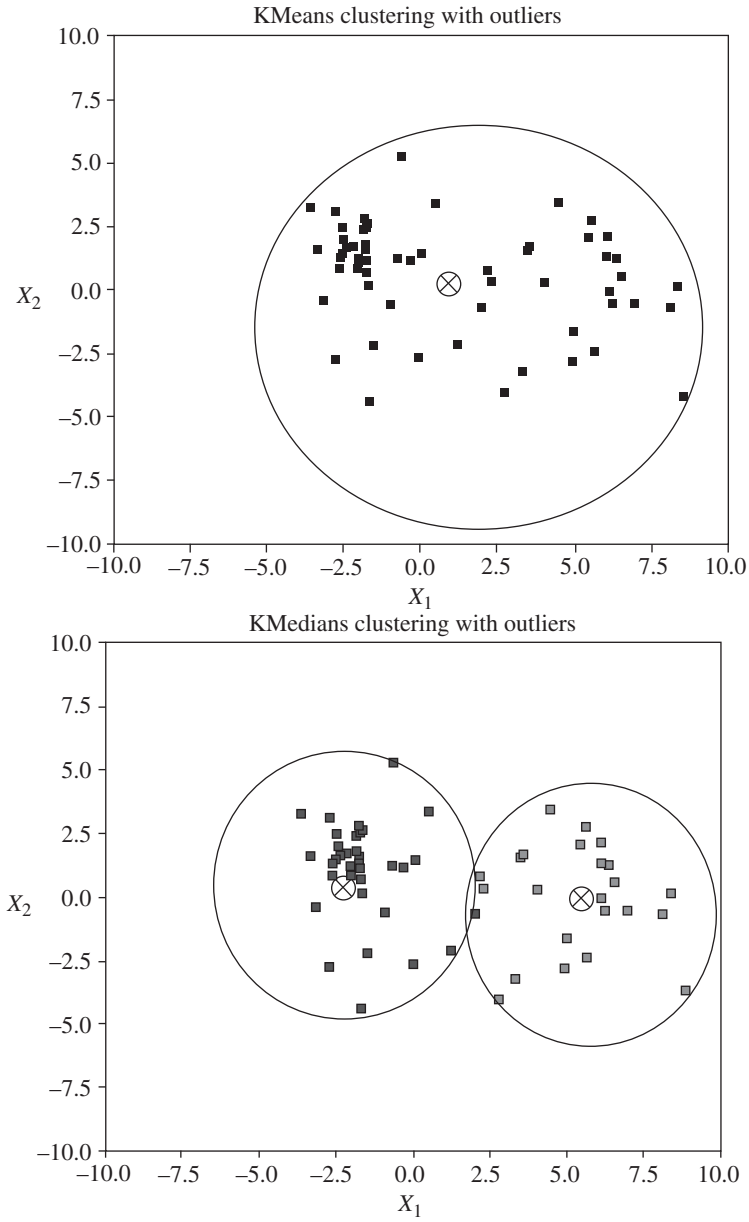


Figure 1.13 *k*-means and *k*-medians clustering with 1 outlier at (−400, −400) which is not shown as it is outside the limits of the plots.

be removed if any $\boldsymbol{x}^{(i)}$ belonging in cluster j satisfies the "3-MADN edit" rule. That is, for all points $\boldsymbol{x}^{(i)}$ in matrix $\boldsymbol{C}_j$,

$$\text{delete } \boldsymbol{x}^{(i)} \quad \text{if } \|\boldsymbol{x}^{(i)} - \text{median}\,(\boldsymbol{C}_j)\|_2 > 3\text{MADN}(\boldsymbol{C}_j) \quad \text{for } i = 1, \ldots, n.$$

This robust version will find and delete all the outliers, regardless of their number and location outside the cluster region.

To compare the two methods, first consider the standard 3-sigma edit rule for the previous dataset with only one outlier. If we use a Gaussian distribution for each cluster, then we find that $\hat{\sigma}_2 = 64.8$ which is too large to be of any value. The 3-sigma edit rule would force us to reject points that are $3 \times 64.8 = 194.4$ units in all directions from the center. It will not delete any points and therefore the 3-sigma statistic is not useful. Contrast this with the robust criteria. The estimate of $\text{MADN}(\boldsymbol{C}_2) = 11.2$ and so the 3-MADN edit rule would yield $3 \times 11.2 = 33.6$. This is a much more reasonable distance to use as a threshold for identifying and deleting outliers, especially since the median is used as the cluster center. This would delete the single outlier in the data. Once this is done, the mean and variance computations will be reasonably accurate. Furthermore, the `KMeans` tool can be used to build the model after the outlier is removed.

Exercise: Given the one-dimensional dataset $\{1, -2, 5, -4, -6, 3\}$ and $K = 2$, compute the centers using k-means. Next, recompute the centers using k-medians.

Ans.: k-means $\{-4, 3\}$. k-medians $\{-4, 3\}$. □

Exercise: Given the one-dimensional dataset $\{1, -2, 5, -4, -66, 3\}$ and $K = 2$, compute the centers using k-means. Next, recompute the centers using k-medians.

Ans.: k-means $\{-24, 3\}$. k-medians $\{-4, 3\}$. □

The 3-MADN edit rule is somewhat arbitrary and may be too strict for most datasets. In practice, we may need to adjust the rule to avoid declaring inliers as outliers. For example, a 10-MADN edit rule may be better to ensure that true outliers are found far from the centers of the clusters. Some tuning may be needed to find the proper multiplier. On the other hand, the 3-sigma rule would fail because the value of $\hat{\sigma}$ would be unreliable. The key point here is that outlier detection requires the use of robust procedures and robust statistics.

1.11 Summary

This chapter highlighted the use of the median as a key to robust methods. It provided a case study of how to robustify an existing algorithm in machine learning and how to detect and remove outliers. It is clear that the k-medians approach is robust to outliers and superior to k-means in this regard. Using k-medians avoids having to find and remove outliers or be concerned with centers that do not represent the data well enough for use in other applications. The k-means method can be safely used only if there

are no outliers in the data. Unfortunately, this assumption is not a realistic for most datasets. Another key point is that the myth that "having lots of data in the dataset tends to reduce the effect of outliers" is dispelled here. Any of the clusters can be affected by just one outlier even if there are many points in the dataset. It is more about the location of the outliers and their effect on cluster centers rather than the number of outliers. Another key point is that robust methods are needed in the detection of outliers and anomalies. This will be a recurring theme throughout the book as we progress through different algorithms that require robustification.

In the rest of this book, we address machine learning through the lens of outliers and anomalies, as opposed to the traditional view which tends to ignore or mishandle them. The initial machine learning example of clustering given in this chapter serves as a template for how we will be covering other methods in this book. However, our focus will shift to the supervised learning tasks of linear regression, logistic regression, and neural networks. The same issues will arise in all these areas, so the techniques themselves will have to be modified to robustify the procedures.

Problems

To complete the projects in this book, you will need a Python programming environment for machine learning. Jupyter notebooks are recommended for running the provided code segments, but you can also use PyCharm or any other Integrated Development Environment (IDE) of your choice. You should set yourself up accordingly at this stage to get the most out of the book.

1.1 a) For the set $S = \{1, 2, 3, 4, 5, 100, 101, 102\}$, find the median and the mean. How many more values above 100 would have to be added to the set before the median is ≥ 100?

b) For the set $S = \{1, 2, 3, 4, 5, 100, 101, 102\}$, compute the standard deviation, SD($\boldsymbol{x}$), and the normalized median absolute deviation, MADN($\boldsymbol{x}$). Assume that we want to remove outliers. Use the 3-sigma and 3-MADN edit rules. Which approach is better at removing the outliers?

c) For the matrix given below

$$\boldsymbol{X} = \begin{pmatrix} 5 & 2 & 7 \\ 1 & 9 & 2 \\ 2 & 1 & 1 \end{pmatrix},$$

compute the column-wise median and mean.

1.2 a) For the vector quantity $\boldsymbol{u} = (-1, -2, 3, 4, 5)^\top$, compute the L_1- and L_2-norms. Which norm is larger? Will this always be true? If not, construct a vector to demonstrate your answer.

b) For the same vector, compute the squared L_2-norm. Again, compare the L_1-norm against the squared L_2-norm. Which is larger and will this always be the case? Provide a counterexample if not true.

c) Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two vectors in $\mathbb{R}^d$. How likely is it that $\|\boldsymbol{v}\|_1 \leq \|\boldsymbol{v}\|_2^2$? Can you find a vector $\boldsymbol{u}$ such that $\|\boldsymbol{u}\|_1 > \|\boldsymbol{u}\|_2^2$? If so, state the vector. If not, explain why.

1.3 A data matrix $\boldsymbol{X} \in \mathbb{R}^{7\times 2}$ is clustered into two groups as follows:

$$\boldsymbol{C}_1 = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ -8 & -8 \end{pmatrix}, \quad \boldsymbol{C}_2 = \begin{pmatrix} 5 & 1 \\ 6 & 1 \\ 6 & 2 \end{pmatrix}$$

a) For k-means, compute the cluster means and total clustering cost.
b) For k-medians, compute the cluster medians and total clustering cost.
c) Plot all the points from both clusters on two graphs, one for k-means and one for k-medians. Indicate the means and medians with $\otimes$. Comment on the results in terms of the tolerance of the two methods with respect to the outlier data $(-8, -8)$.
d) Compute the covariance matrix, $\sigma^2 \boldsymbol{I}$ of each cluster. This requires only the estimation of σ^2 for each cluster. Use the mean values computed above. For σ_1^2 in cluster 1, compute matrix $\boldsymbol{C}_1 - \boldsymbol{\mu}_1$, then average the sum the squares of all terms. In Python, `sigma12 = np.mean ((C1-mu1)**2)` would produce the estimate. The same procedure can be used for cluster 2.
e) Estimate π_1 and π_2 using a ratio of the number of points in a cluster to the total number of points in the dataset. Write an expression for the model of the Gaussian mixture. Is this a good model?
f) How would you use the median and MADN to build a better model? Write the expression for the better model.

1.4 (PROJECT) In this project, the objective is to create a dataset for clustering using Gaussian mixtures.
a) Import the needed libraries.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from sklearn.cluster import KMeans
```

b) Create a dataset for the Gaussian mixture given by

$$f_X(x) = 0.6\,\mathcal{N}_2((-2,1)^\top, 3\boldsymbol{I}_2) + 0.4\,\mathcal{N}_2((5,0)^\top, 4\boldsymbol{I}_2).$$

The data can be generated and plotted with the following code:

```
np.random.seed(8)
mu = np.array([[-2,1],[5,0]])
sigma = np.array([[[3,0],[0,3]],[[4,0],[0,4]]])
Y1 =  np.random.multivariate_normal(mu[0], \
    sigma[0], size=60)
Y2 =  np.random.multivariate_normal(mu[1], \
    sigma[1], size=40)
X = np.vstack((Y1,Y2))
plt.xlim([-10,10])
plt.ylim([-10,10])
```

```
plt.scatter(X[:,0],X[:,1], \
    marker="$.$",color="black")
plt.show()
```

1.5 (PROJECT) The goal of this project is to compare `KMeans` against `KMedians` using the same dataset as in the project above.

a) Run the `KMeans` program in the `sklearn.cluster` library for the dataset *without any outliers* using:

```
kmeans = KMeans(n_clusters=2, init="k-means++", \
        random_state=1).fit(X)
labels = kmeans.labels_
means = kmeans.cluster_centers_
print("Centers from KMeans")
print(means)
```

b) The Python code given below, called `KMedians`, generates robust cluster centers given $\boldsymbol{X}$ and the number of clusters, K. Enter and run it in your Python notebook.

```
# Assign points to clusters
def assign_clusters(X, M):
    distances = np.sum(np.abs(X[:,  \
        np.newaxis] - M), axis=2)
    clusters = np.argmin(distances, axis=1)
    return clusters

# Find the cluster centers
def calculate_cluster_medians(X, clusters, \
        num_clusters):
    cluster_medians = np.zeros((num_clusters, \
        X.shape[1]))
    for i in range(num_clusters):
        cluster_rows = X[clusters == i]
        if len(cluster_rows) > 0:
            cluster_medians[i] = \
                np.median(cluster_rows, axis=0)
    return cluster_medians

# Compute the cost of this arrangement
def calculate_total_cost(X, clusters, \
    cluster_medians):
    total_cost = 0
    for i in range(len(X)):
```

```
        cluster_median = \
        cluster_medians[clusters[i]]
        total_cost += np.sum(np.abs(X[i] \
            - cluster_median))
    return total_cost

# Plot 2D cluster arrangement
def plot_clusters(X, clusters, cluster_medians):
    plt.figure(figsize=(10, 6))
    num_clusters = cluster_medians.shape[0]
    # Plot each cluster in a different color
    for i in range(num_clusters):
        cluster_points = X[clusters == i]
        plt.scatter(cluster_points[:, 0], \
                    cluster_points[:, 1], \
                    label=f'cluster i}')
    # Plot the cluster medians
    plt.scatter(cluster_medians[:, 0], \
                cluster_medians[:, 1], \
                color='black', marker='x', \
                s=100, label='cluster medians')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('clusters and their medians')
    plt.legend()
    plt.show()

# Perform k-medians clustering
def KMedians(X,K):
    seed = 0
    np.random.seed(seed)
    n, d = X.shape
    M = X[np.random.choice(n, K, replace=False)]
    prev_cost = None
    new_cost = None
    #Use Lloyd-style iteration
    while (prev_cost is None or \
        np.abs(prev_cost - new_cost) > 1e-4):
        clusters = assign_clusters(X, M)
        num_clusters = M.shape[0]
        M = calculate_cluster_medians(X, \
        clusters, num_clusters)
```

```
        prev_cost = new_cost
        new_cost = calculate_total_cost(X, \
        clusters, M)
    return clusters,new_cost, M
```

c) Run KMedians using the dataset $\boldsymbol{X}$ generated previously and plot the results. How do the two methods compare in terms of computing the centers of the clusters?

```
K = 2
cluster, cost, medians = KMedians(X,K)
print("Centers from KMedians")
print(medians)
plot_clusters(X, cluster, medians)
```

d) Use the code below to plot the results of the `KMeans` and `KMedians` programs next to each other as subplots.

```
X1md = np.array(X[cluster == 1])
X2md = np.array(X[cluster == 0])
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
axes[0].scatter(X[:, 0], X[:, 1], c=labels, \
            cmap="plasma", marker="$.$", \
            edgecolors="k", s=50)
axes[0].set_title("KMeans Clustering")
axes[0].set_xlabel("Feature 1")
axes[0].set_ylabel("Feature 2")
axes[0].scatter(means[0,0],means[0,1], \
    marker="x",color="red")
axes[0].scatter(means[1,0],means[1,1], \
    marker="x",color="red")
axes[0].set_xlim([-10,10])
axes[0].set_ylim([-10,10])
axes[1].scatter(X1md[:, 0], X1md[:, 1], c="yellow",
        marker="$.$", edgecolors="k", s=50)
axes[1].scatter(X2md[:, 0], X2md[:, 1], c="black",
        marker="$.$",edgecolors="k", s=50)
axes[1].set_title("KMedians Clustering")
axes[1].set_xlabel("Feature 1")
axes[1].set_ylabel("Feature 2")
plt.scatter(medians[0,0],medians[0,1], \
    marker="x",color="red")
plt.scatter(medians[1,0],medians[1,1], \
    marker="x",color="red")
axes[1].set_xlim([-10,10])
```

```
axes[1].set_ylim([-10,10])

plt.tight_layout()
plt.show()
```

What do you notice about the cluster centers in each case?

e) Next, insert outliers using the code below and then rerun both methods. For `KMeans`, use

```
X[1] = np.array([-400,-400])
X[2] = np.array([-400,0])
kmeans = KMeans(n_clusters=2,  init="k-means++", \
        random_state=1).fit(X)
labels = kmeans.labels_
means = kmeans.cluster_centers_
print("New centers from KMeans")
print(means)
```

For `KMedians`, use

```
cluster, cost, medians = KMedians(X,K)
print("New centers from KMedians")
print(medians)
```

Compare `KMedians` and `KMeans` on the dataset *with outliers*.

f) Use the same plotting code as in part (c) to plot the `KMeans` and `KMedians` results again showing the clusters in each case after outliers are inserted. How many cluster centers can you find in each case?

1.6 (PROJECT) In this project, the objective is to detect outliers in the dataset from the previous project. To do this, we need to use the `KMedians` results and the 10-MADN edit rule. Which outliers are found using the code below?

```
# Outlier detection method using MADN
X1md = np.array(X[cluster == 1])
X2md = np.array(X[cluster == 0])
def MADN(r):
    return(1.4826*np.median(np.abs(r-np.median(r))))
dist1 = (np.abs((X1md - medians[0,:]))).sum(axis=1)
mad1 = MADN(dist1)
dist2 = (np.abs((X2md - medians[1,:]))).sum(axis=1)
mad2 = MADN(dist2)
for i in range(X1md.shape[0]):
    if (np.sum(np.abs(X1md[i, :] - medians[0,:])) \
        > 10*mad1):
        print("Outlier found at",X1md[i,:]," \
```

```
                in Cluster 1")
    for i in range(X2md.shape[0]):
        if (np.sum(np.abs(X2md[i, :] - medians[1,:])) \
            > 10*mad2):
            print("Outlier found at",X2md[i,:]," \
                in Cluster 2")
```

References

James, G., Witten, D., Hastie, T. et al. (2023). *An Introduction to Statistical Learning with Applications in Python*. Springer.

Maronna, R.A., Douglas Martin, R., Yohai, V.J., and Salibian-Barrera, M. (2019). *Robust Statistics, Theory and Methods (with R)*. Wiley.