Chapter 1

An Introduction to Outlier Analysis

> "Never take the comment that you are different as a condemnation, it might be a compliment. It might mean that you possess unique qualities that, like the most rarest of diamonds is ... one of a kind." – Eugene Nathaniel Butler

1.1 Introduction

An outlier is a data point that is significantly different from the remaining data. Hawkins defined [249] an outlier as follows:

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

Outliers are also referred to as *abnormalities*, *discordants*, *deviants*, or *anomalies* in the data mining and statistics literature. In most applications, the data is created by one or more generating processes, which could either reflect activity in the system or observations collected about entities. When the generating process behaves unusually, it results in the creation of outliers. Therefore, an outlier often contains useful information about abnormal characteristics of the systems and entities that impact the data generation process. The recognition of such unusual characteristics provides useful application-specific insights. Some examples are as follows:

- **Intrusion detection systems:** In many computer systems, different types of data are collected about the operating system calls, network traffic, or other user actions. This data may show unusual behavior because of malicious activity. The recognition of such activity is referred to as intrusion detection.

- **Credit-card fraud:** Credit-card fraud has become increasingly prevalent because of greater ease with which sensitive information such as a credit-card number can be compromised. In many cases, unauthorized use of a credit card may show different patterns, such as buying sprees from particular locations or very large transactions. Such patterns can be used to detect outliers in credit-card transaction data.

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_1

- **Interesting sensor events:** Sensors are often used to track various environmental and location parameters in many real-world applications. Sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks. As discussed later in this book, event detection is an important *temporal* version of outlier detection.

- **Medical diagnosis:** In many medical applications, the data is collected from a variety of devices such as magnetic resonance imaging (MRI) scans, positron emission tomography (PET) scans or electrocardiogram (ECG) time-series. Unusual patterns in such data typically reflect disease conditions.

- **Law enforcement:** Outlier detection finds numerous applications in law enforcement, especially in cases where unusual patterns can only be discovered over time through multiple actions of an entity. Determining fraud in financial transactions, trading activity, or insurance claims typically requires the identification of unusual patterns in the data generated by the actions of the criminal entity.

- **Earth science:** A significant amount of spatiotemporal data about weather patterns, climate changes, or land-cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about human activities or environmental trends that may be the underlying causes.

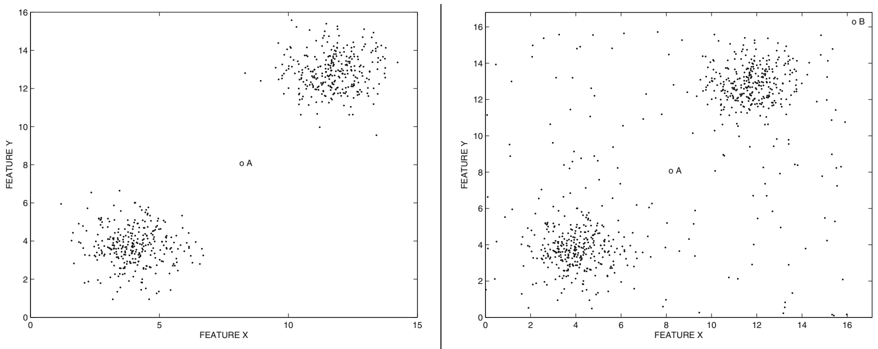
In all these applications, the data has a “normal” model, and anomalies are recognized as deviations from this normal model. Normal data points are sometimes also referred to as *inliers*. In some applications such as intrusion or fraud detection, outliers correspond to *sequences* of multiple data points rather than individual data points. For example, a fraud event may often reflect the actions of an individual in a particular sequence. The specificity of the sequence is relevant to identifying the anomalous event. Such anomalies are also referred to as *collective anomalies*, because they can only be inferred collectively from a set or sequence of data points. Such collective anomalies are often a result of unusual *events* that generate anomalous patterns of activity. This book will address these different types of anomalies.

The output of an outlier detection algorithm can be one of two types:

- **Outlier scores:** Most outlier detection algorithms output a score quantifying the level of “outlierness” of each data point. This score can also be used to rank the data points in order of their outlier tendency. This is a very general form of output, which retains all the information provided by a particular algorithm, but it does not provide a concise summary of the small number of data points that should be considered outliers.

- **Binary labels:** A second type of output is a binary label indicating whether a data point is an outlier or not. Although some algorithms might directly return binary labels, outlier scores can also be converted into binary labels. This is typically achieved by imposing thresholds on outlier scores, and the threshold is chosen based on the statistical distribution of the scores. A binary labeling contains less information than a scoring mechanism, but it is the final result that is often needed for decision making in practical applications.

It is often a subjective judgement, as to what constitutes a “sufficient” deviation for a point to be considered an outlier. In real applications, the data may be embedded in a



(a) No noise

(b) With noise

Figure 1.1: The difference between noise and anomalies

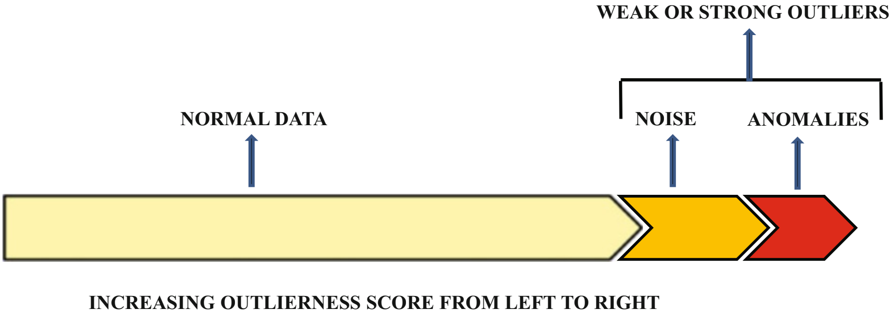


Figure 1.2: The spectrum from normal data to outliers

significant amount of noise, and such noise may not be of any interest to the analyst. It is usually the *significantly interesting deviations* that are of interest. In order to illustrate this point, consider the examples illustrated in Figures 1.1(a) and (b). It is evident that the main patterns (or clusters) in the data are identical in both cases, although there are significant differences outside these main clusters. In the case of Figure 1.1(a), a single data point (marked by ‘A’) seems to be very different from the remaining data, and is therefore very obviously an anomaly. The situation in Figure 1.1(b) is much more subjective. While the corresponding data point ‘A’ in Figure 1.1(b) is also in a sparse region of the data, it is much harder to state confidently that it represents a true deviation from the remaining data set. It is quite likely that this data point represents randomly distributed noise in the data. This is because the point ‘A’ seems to fit a pattern represented by other randomly distributed points. Therefore, throughout this book, the term “outlier” refers to a data point that could either be considered an abnormality or noise, whereas an “anomaly” refers to a special kind of outlier that is of interest to an analyst.

In the *unsupervised scenario*, where previous examples of interesting anomalies are not available, the noise represents the semantic boundary between normal data and true anomalies– noise is often modeled as a weak form of outliers that does not always meet the strong criteria necessary for a data point to be considered interesting or anomalous enough. For example, data points at the boundaries of clusters may often be considered noise. Typically, most outlier detection algorithms use some quantified measure of the *outlierness* of a data point, such as the sparsity of the underlying region, nearest neighbor based distance, or the fit to the underlying data distribution. Every data point lies on a continuous spectrum from normal data to noise, and finally to anomalies, as illustrated in Figure 1.2. The separation of the different regions of this spectrum is often not precisely defined, and is chosen on an *ad hoc* basis according to application-specific criteria. Furthermore, the separation between noise and anomalies is not pure, and many data points created by a noisy generative process may be deviant enough to be interpreted as anomalies on the basis of the outlier score. Thus, anomalies will *typically* have a much higher outlier score than noise, but this is not a distinguishing factor between the two as a matter of *definition.* Rather, it is the interest of the analyst that regulates the distinction between noise and an anomaly.

Some authors use the terms *weak outliers* and *strong outliers* in order to distinguish between noise and anomalies [4, 318]. The detection of noise in the data has numerous applications of its own. For example, the removal of noise creates a much cleaner data set, which can be utilized for other data mining algorithms. Although noise might not be interesting in its own right, its *removal and identification* continues to be an important problem for mining purposes. Therefore, both noise and anomaly detection problems are important enough to be addressed in this book. Throughout this book, methods specifically relevant to *either* anomaly detection or noise removal will be identified. However, the bulk of the outlier detection algorithms could be used for either problem, since the difference between them is really one of semantics.

Since the semantic distinction between noise and anomalies is based on analyst interest, the best way to find such anomalies and distinguish them from noise is to use the feedback from *previously known outlier examples of interest.* This is quite often the case in many applications, such as credit-card fraud detection, where previous examples of interesting anomalies may be available. These may be used in order to learn *a model that distinguishes the normal patterns from the abnormal data.* Supervised outlier detection techniques are typically much more effective in many application-specific scenarios, because the characteristics of the previous examples can be used to sharpen the search process towards more relevant outliers. This is important, because outliers can be defined in numerous ways in a given data set, most of which may not be interesting. For example, in Figures 1.1(a) and (b), previous examples may suggest that only records with unusually high values of both attributes should be considered anomalies. In such a case, the point ‘A’ in *both* figures should be regarded as noise, and the point ‘B’ in Figure 1.1(b) should be considered an anomaly instead! The crucial point to understand here is that anomalies need to be *unusual in an interesting way*, and the supervision process re-defines what one might find interesting. Generally, unsupervised methods can be used either for noise removal or anomaly detection, and supervised methods are designed for application-specific anomaly detection. Unsupervised methods are often used in an exploratory setting, where the discovered outliers are provided to the analyst for further examination of their application-specific importance.

Several levels of supervision are possible in practical scenarios. In the fully supervised scenario, examples of both normal and abnormal data are available that can be clearly distinguished. In some cases, examples of outliers are available, but the examples of “normal” data may also contain outliers in some (unknown) proportion. This is referred to as classification with positive and unlabeled data. In other semi-supervised scenarios, only examples of normal data or only examples of anomalous data may be available. Thus, the number of variations of the problem is rather large, each of which requires a related but dedicated set of techniques.

Finally, the data representation may vary widely across applications. For example, the data may be purely multidimensional with no relationships among points, or the data may be sequential with temporal ordering, or may be defined in the form of a network with arbitrary relationships among data points. Furthermore, the attributes in the data may be numerical, categorical, or mixed. Clearly, the outlier detection process needs to be sensitive to the nature of the attributes and relationships in the underlying data. In fact, the relationships themselves may often provide an outlier-detection criterion in the form of connections between entities that do not usually occur together. Such outliers are referred to as *contextual* outliers. A classical example of this is the concept of *linkage outliers* in social network analysis [17]. In this case, entities (nodes) in the graph that are normally not connected together may show *anomalous* connections with each other. Thus, the impact of data types on the anomaly detection process is significant and will be carefully addressed in this book.

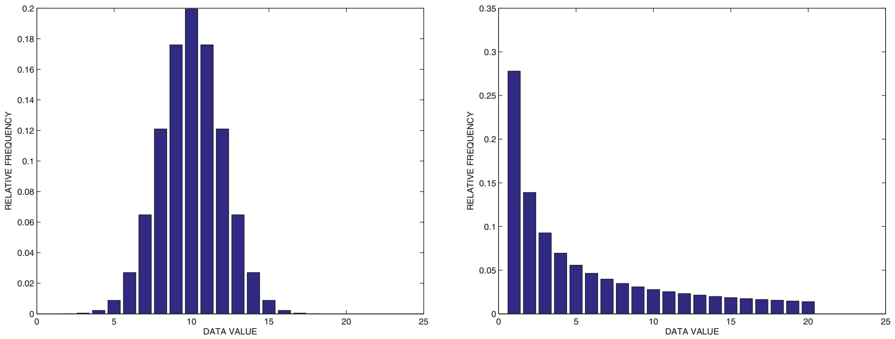
This chapter is organized as follows. In section 1.2, the importance of data modeling in outlier analysis is discussed. In section 1.3, the basic outlier models for outlier detection are introduced. Outlier ensembles are introduced in section 1.4. Section 1.5 discusses the basic data types used for analysis. Section 1.6 introduces the concept of supervised modeling of outliers for data analysis. Methods for evaluating outlier detection algorithms are discussed in section 1.7. The conclusions are presented in section 1.8.

1.2 The Data Model is Everything

Virtually all outlier detection algorithms create a model of the normal patterns in the data, and then compute an outlier score of a given data point on the basis of the deviations from these patterns. For example, this data model may be a generative model such as a Gaussian-mixture model, a regression-based model, or a proximity-based model. All these models make different assumptions about the "normal" behavior of the data. The outlier score of a data point is then computed by evaluating the quality of the fit between the data point and the model. In many cases, the model may be algorithmically defined. For example, nearest neighbor-based outlier detection algorithms model the outlier tendency of a data point in terms of the distribution of its k-nearest neighbor distance. Thus, in this case, the assumption is that outliers are located at large distances from most of the data.

Clearly, the choice of the data model is crucial. An incorrect choice of data model may lead to poor results. For example, a fully generative model such as the Gaussian mixture model may not work well, if the data does not fit the generative assumptions of the model, or if a sufficient number of data points are not available to learn the parameters of the model. Similarly, a linear regression-based model may work poorly, if the underlying data is clustered arbitrarily. In such cases, data points may be incorrectly reported as outliers *because of poor fit to the erroneous assumptions of the model.* Unfortunately, outlier detection is largely an *unsupervised* problem in which examples of outliers are not available to learn[1] the best model (in an automated way) for a particular data set. This aspect of outlier detection tends to make it more challenging than many other *supervised* data mining problems like classification in which *labeled* examples are available. Therefore, in practice, the choice of the model is often dictated by the analyst's understanding of the kinds of deviations relevant to an application. For example, in a spatial application measuring a behavioral attribute such as the location-specific temperature, it would be reasonable to assume that an unusual deviation of the temperature attribute in a spatial locality is an

[1] In supervised problems like classification, this process is referred to as *model selection.*



(a) Normal distribution (b) Zipf distribution

Figure 1.3: Applying Z-value test on the Normal and Zipf distributions

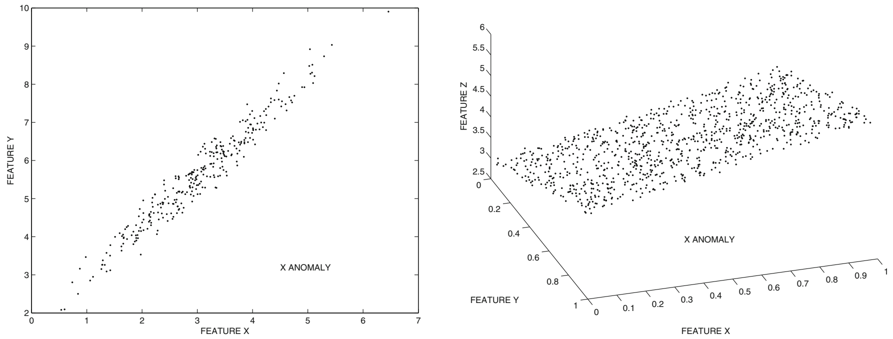
indicator of abnormality. On the other hand, for the case of high-dimensional data, even the definition of data locality may be ill-defined because of data sparsity. Thus, an effective model for a particular data domain may only be constructed after carefully evaluating the relevant modeling properties of that domain.

In order to understand the impact of the model, it is instructive to examine the use of a simple model known as the *Z-value test* for outlier analysis. Consider a set of 1-dimensional quantitative data observations, denoted by $X_1 \dots X_N$, with mean μ and standard deviation σ. The Z-value for the data point X_i is denoted by Z_i and is defined as follows:

$$Z_i = \frac{|X_i - \mu|}{\sigma} \tag{1.1}$$

The Z-value test computes the number of standard deviations by which a data point is distant from the mean. This provides a good proxy for the outlier score of that point. An implicit assumption is that the data is modeled from a normal distribution, and therefore the Z-value is a random variable drawn from a *standard* normal distribution with zero mean and unit variance. In cases where the mean and standard deviation of the distribution can be accurately estimated, a good "rule-of-thumb" is to use $Z_i \geq 3$ as a proxy for the anomaly. However, in scenarios in which very few samples are available, the mean and standard deviation of the underlying distribution cannot be estimated robustly. In such cases, the results from the Z-value test need to be interpreted more carefully with the use of the (related) *Student's t-distribution* rather than a normal distribution. This issue will be discussed in Chapter 2.

It is often forgotten by practitioners during modeling that the Z-value test implicitly assumes a normal distribution for the underlying data. When such an approximation is poor, the results are harder to interpret. For example, consider the two data frequency histograms drawn on values between 1 and 20 in Figure 1.3. In the first case, the histogram is sampled from a normal distribution with $(\mu, \sigma) = (10, 2)$, and in the second case, it is sampled from a Zipf distribution $1/i$. It is evident that most of the data lies in the range $[10 - 2 * 3, 10 + 2 * 3]$ for the normal distribution, and all data points lying outside this range can be truly considered anomalies. Thus, the Z-value test works very well in this case. In the second case with the Zipf distribution, the anomalies are not quite as clear, although the data points with very high values (such as 20) can probably be considered anomalies. In this case, the mean and standard deviation of the data are 5.24 and 5.56, respectively. As a result, the Z-value test does not declare *any* of the data points as anomalous (for a



(a) 2-d data (b) 3-d data

Figure 1.4: Linearly Correlated Data

threshold of 3), although it does come close. In any case, the significance of the Z-value from the Zipf-distribution is not very meaningful at least from the perspective of probabilistic interpretability. This suggests that if mistakes are made at the modeling stage, it can result in an incorrect understanding of the data. Such tests are often used as a *heuristic* to provide a rough idea of the outlier scores even for data sets that are far from normally distributed, and it is important to interpret such scores carefully.

An example in which the Z-value test would not work even as a heuristic, would be one in which it was applied to a data point that was an outlier only because of its relative position, rather than its extreme position. For example, if the Z-value test is applied to an individual dimension in Figure 1.1(a), the test would fail miserably, because point 'A' would be considered the most centrally located and normal data point. On the other hand, the test can still be reasonably applied to a set of *extracted* 1-dimensional values corresponding to the k-nearest neighbor distances of each point. Therefore, the effectiveness of a model depends both on the choice of the test used, and *how* it is applied.

The best choice of a model is often data-specific. This requires a good understanding of the data itself before choosing the model. For example, a regression-based model would be most suitable for finding the outliers in the data distributions of Figure 1.4, where most of the data is distributed along linear correlation planes. On the other hand, a clustering model would be more suitable for the cases illustrated in Figure 1.1. An poor choice of model for a given data set is likely to provide poor results. *Therefore, the core principle of discovering outliers is based on assumptions about the structure of the normal patterns in a given data set. Clearly, the choice of the "normal" model depends highly on the analyst's understanding of the natural data patterns in that particular domain.* This implies that it is often useful for the analyst to have a semantic understanding of the data representation, although this is often not possible in real settings.

There are many trade-offs associated with model choice; a highly complex model with too many parameters will most likely overfit the data, and will also find a way to fit the outliers. A simple model, which is constructed with a good intuitive understanding of the data (and possibly also an understanding of what the analyst is looking for), is likely to lead to much better results. On the other hand, an oversimplified model, which fits the data poorly, is likely to declare normal patterns as outliers. The initial stage of selecting the data model is perhaps the most crucial one in outlier analysis. The theme about the impact of data models will be repeated throughout the book, with specific examples.

1.2.1 Connections with Supervised Models

One can view the outlier detection problem as a variant of the classification problem in which the class label ("normal" or "anomaly") is unobserved. Therefore, by virtue of the fact that normal examples far outnumber the anomalous examples, one can "pretend" that the entire data set contains the normal class and create a (possibly noisy) model of the normal data. Deviations from the normal model are treated as outlier scores. This connection between classification and outlier detection is important because much of the theory and methods from classification generalize to outlier detection [32]. The unobserved nature of the labels (or outlier scores) is the reason that outlier detection methods are referred to as *unsupervised* whereas classification methods are referred to as *supervised*. In cases where the anomaly labels are observed, the problem simplifies to the imbalanced version of data classification, and it is discussed in detail in Chapter 7.

The model of normal data for unsupervised outlier detection may be considered a *one-class analog* of the multi-class setting in classification. However, the one-class setting is sometimes far more subtle from a modeling perspective, because it is much easier to distinguish between examples of two classes than to predict whether a particular instance matches examples of a single (normal) class. When at least two classes are available, the *distinguishing* characteristics between the two classes can be learned more easily in order to sharpen the accuracy of the model.

In many forms of predictive learning, such as classification and recommendation, there is a natural dichotomy between *instance-based learning methods* and *explicit generalization methods*. Since outlier detection methods require the design of a model of the normal data in order to make predictions, this dichotomy applies to the unsupervised domain as well. In instance-based methods, a training model is not constructed up front. Rather, for a given test instance, one computes the most relevant (i.e., closest) instances of the training data, and makes predictions on the test instance using these related instances. Instance-based methods are also referred to as *lazy learners* in the field of classification [33] and *memory-based methods* in the field of recommender systems [34].

A simple example of an instance-based learning method in outlier analysis is the use of the 1-nearest-neighbor distance of a data point as its outlier score. Note that this approach does not require the construction of a training model up front because all the work of determining the nearest neighbor is done after specifying the identity of the instance to be predicted (scored). The 1-nearest neighbor outlier detector can be considered the unsupervised analog of the 1-nearest neighbor classifier in the supervised domain. Instance-based models are extremely popular in the outlier analysis domain because of their simplicity, effectiveness, and intuitive nature. In fact, many of the most popular and successful methods for outlier detection, such as k-nearest neighbor detectors [58, 456] and Local Outlier Factor (LOF) [96] (cf. Chapter 4), are instance-based methods.

The popularity of instance-based methods is so great in the outlier analysis community that the vast array of one-class analogs of other supervised methods are often overlooked. In principle, almost any classification method can be re-designed to create a one-class analog. Most of these methods are *explicit generalization methods*, in which a summarized model needs to be created up front. Explicit generalization methods use a two-step process on the data set $\mathcal{D}$:

1. Create a one-class model of the normal data using the original data set $\mathcal{D}$. For example, one might learn a linear hyperplane describing the normal data in Figure 1.4(b). This hyperplane represents a *summarized model* of the entire data set and therefore represents an explicit generalization of the data set.

Table 1.1: Classification methods and their unsupervised analogs in outlier analysis

Supervised Model	**Unsupervised Analog(s)**	**Type**
k-nearest neighbor	k-NN distance, LOF, LOCI (Chapter 4)	Instance-based
Linear Regression	Principal Component Analysis (Chapter 3)	Explicit Generalization
Naive Bayes	Expectation-maximization (Chapter 2)	Explicit Generalization
Rocchio	Mahalanobis method (Chapter 3) Clustering (Chapter 4)	Explicit Generalization
Decision Trees Random Forests	Isolation Trees Isolation Forests (Chapters 5 and 6)	Explicit generalization
Rule-based	FP-Outlier (Chapter 8)	Explicit Generalization
Support-vector machines	One-class support-vector machines (Chapter 3)	Explicit generalization
Neural Networks	Replicator neural networks (Chapter 3)	Explicit generalization
Matrix factorization (incomplete data prediction)	Principal component analysis Matrix factorization (Chapter 3)	Explicit generalization

2. Score each point in $\mathcal{D}$ based on its deviation from this model of normal data. For example, if we learn a linear hyperplane using the data set of Figure 1.4(b) in the first step, then we might report the Euclidean distance from this hyperplane as the outlier score.

One problem with explicit generalization methods is that the same data set $\mathcal{D}$ is used for both training and scoring. This is because it is hard to exclude a specific test point during the scoring process (like instance-based methods). Furthermore, unlike classification in which the presence or absence of ground-truth (labeling) naturally partitions the data into training and test portions, there is no labeling available in unsupervised problems. Therefore, one typically wants to use the entire data set $\mathcal{D}$ for both training and testing in unsupervised problems, which causes overfitting. Nevertheless, the influence of individual points on overfitting is often small in real-world settings because explicit generalization methods tend to create a concise summary (i.e., *generalized* representation) of a much larger data set. Since the same data $\mathcal{D}$ is used for training and testing, one can view outlier scores as the training data errors made in the assumption of "pretending" that all training data points belong to the normal class. Often an effective approach to reduce overfitting is to repeatedly partition the data into training and test sets in a randomized way and average the outlier scores of test points from the various models. Such methods will be discussed in later chapters.

Virtually all classification models can be generalized to outlier detection by using an appropriate one-class analog. Examples of such models include linear regression models, principal component analysis, probabilistic expectation-maximization models, clustering methods, one-class support vector machines, matrix factorization models, and one-class neural networks. For the reader who has a familiarity with the classification problem, we have listed various classification models and their corresponding one-class analogs for outlier detection in Table 1.1. The table is not comprehensive and is intended to provide intuition about the connections between the supervised and unsupervised settings with representative examples. The connections between supervised and unsupervised learning are very deep; in section 7.7 of Chapter 7, we point out yet another useful connection between outlier detection and regression modeling. This particular connection has the merit that it enables the use of hundreds of off-the-shelf regression models for unsupervised outlier detection with an almost trivial implementation.

1.3 The Basic Outlier Detection Models

This section will present an overview of the most important models in the literature, and also provide some idea of the settings in which they might work well. A detailed discussion of these methods are provided in later chapters. Several factors influence the choice of an outlier model, including the data type, data size, availability of relevant outlier examples, and the need for interpretability in a model. The last of these criteria merits further explanation.

The *interpretability* of an outlier detection model is extremely important from the perspective of the analyst. It is often desirable to determine *why* a particular data point should be considered an outlier because it provides the analyst further hints about the diagnosis required in an application-specific scenario. This process is also referred to as that of discovering the *intensional knowledge* about the outliers [318] or that of *outlier detection and description* [44]. Different models have different levels of interpretability. Typically, models that work with the original attributes and use fewer transforms on the data (e.g., principal component analysis) have higher interpretability. The trade-off is that data transformations often enhance the contrast between the outliers and normal data points at the expense of interpretability. Therefore, it is critical to keep these factors in mind while choosing a specific model for outlier analysis.

1.3.1 Feature Selection in Outlier Detection

It is notoriously difficult to perform feature selection in outlier detection because of the unsupervised nature of the outlier detection problem. Unlike classification, in which labels can be used as guiding posts, it is difficult to learn how features relate to the (unobserved) ground truth in unsupervised outlier detection. Nevertheless, a common way of measuring the non-uniformity of a set of univariate points $x_1 \ldots x_N$ is the *Kurtosis measure.* The first step is to compute the mean μ and standard deviation σ of this set of values and standardize the data to zero mean and unit variance as follows:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{1.2}$$

Note that the mean value of the *squares* of z_i is always 1 because of how z_i is defined. The Kurtosis measure computes the mean value of the *fourth* power of z_i:

$$K(z_1 \ldots z_N) = \frac{\sum_{i=1}^{N} z_i^4}{N} \tag{1.3}$$

Feature distributions that are very non-uniform show a high level of Kurtosis. For example, when the data contains a few extreme values, the Kurtosis measure will increase because of the use of the fourth power. Kurtosis measures are often used [367] in the context of *subspace outlier detection methods* (see Chapter 5), in which outliers are explored in lower-dimensional projections of the data.

One problem with the Kurtosis measure is that it does not use the *interactions* between various attributes well, when it analyzes the features individually. It is also possible to use the Kurtosis measure on lower-dimensional distance distributions. For example, one can compute the Kurtosis measure on the set of N Mahalanobis distances of all data points to the centroid of the data after the data has been projected into a lower-dimensional subspace S. Such a computation provides the *multidimensional* Kurtosis of that subspace S, while taking into account the interactions between the various dimensions of S. The Mahalanobis distance is introduced in Chapter 2. One can combine this computation with a greedy method of iteratively adding features to a candidate subset S of features in order to construct a discriminative subset of dimensions with the highest multidimensional Kurtosis.

A second methodology for feature selection [429] is to use the connections of the outlier detection problem to supervised learning. The basic idea is that features that are uncorrelated with all other features should be considered irrelevant because outliers often correspond to violation of the model of normal data dependencies. Uncorrelated features cannot be used to model data dependencies. Therefore, if one uses a regression model to predict one of the features from the other features, and the *average* squared error is too large, then such a feature should be pruned. All features are standardized to unit variance and the root-mean squared error $RMSE_k$ of predicting the kth feature from other features is computed. Note that if $RMSE_k$ is larger than 1, then the error of prediction is greater than the feature variance and therefore the kth feature should be pruned. One can also use this approach to weight the features. Specifically, the weight of the kth feature is given by $\max\{0, 1 - RMSE_k\}$. Details of this model are discussed in section 7.7 of Chapter 7.

1.3.2 Extreme-Value Analysis

The most basic form of outlier detection is extreme-value analysis of 1-dimensional data. These are very specific types of outliers in which it is assumed that the values that are either too large or too small are outliers. Such special kinds of outliers are also important in many application-specific scenarios.

The key is to determine the *statistical tails of the underlying distribution.* As illustrated earlier in Figure 1.3, the nature of the tails may vary considerably depending upon the underlying data distribution. The normal distribution is the easiest to analyze, because most statistical tests (such as the Z-value test) can be interpreted directly in terms of probabilities of significance. Nevertheless, even for arbitrary distributions, such tests provide a good heuristic idea of the outlier scores of data points, even when they cannot be interpreted statistically. The problem of determining the tails of distributions has been widely studied in the statistics literature. Details of such methods will be discussed in Chapter 2.

Extreme-value statistics [437] is distinct from the traditional definition of outliers. The traditional definition of outliers, as provided by Hawkins, defines such objects by their *generative probabilities* rather than the extremity in their values. For example, in the data set $\{1, 2, 2, 50, 98, 98, 99\}$ of 1-dimensional values, the values 1 and 99 could, very mildly, be considered extreme values. On the other hand, the value 50 is the average of the data set, and is most definitely not an extreme value. However, the value 50 is isolated from most of the other data values, which are grouped into small ranges such as $\{1, 2, 2\}$ and $\{98, 98, 99\}$. Therefore, most probabilistic and density-based models would classify the value 50 as the strongest outlier in the data, and this result would also be consistent with Hawkins's generative definition of outliers. Confusions between extreme-value analysis and outlier analysis are common, especially in the context of multivariate data. This is quite often the case, since many extreme-value models also use probabilistic models in order to quantify the probability that a data point is an extreme value.

Although extreme-value analysis is naturally designed for univariate (one-dimensional) data, it is also possible to generalize it to multivariate data, by determining the points at the multidimensional *outskirts* of the data. It is important to understand that such outlier detection methods are tailored to determining *specific kinds of* outliers even in the multivariate case. For example, the point 'A' in both Figures 1.1(a) and (b) will not be deemed an extreme value by such methods, since it does not lie on the outer boundary of the data, even though it is quite clearly an outlier in Figure 1.1(a). On the other hand, the point 'B' in Figure 1.1(b) can be considered an extreme value, because it lies on the outskirts of the multidimensional data set.

Extreme-value modeling plays an important role in most outlier detection algorithms as a final step. *This is because most outlier modeling algorithms quantify the deviations of the data points from the normal patterns in the form of a numerical score.* Extreme-value analysis is usually required as a final step on these modeled deviations, since they are now represented as univariate values in which extreme values correspond to outliers. In many multi-criteria outlier detection algorithms, a vector of outlier scores may be obtained (such as extreme values of temperature and pressure in a meteorological application). In such cases, multivariate extreme-value methods can help in *unifying* these outlier scores into a single value, and also generate a binary label output. Therefore, even though the original data may not be in a form where extreme-value analysis is directly helpful, it remains an integral part of the outlier detection process. Furthermore, many real-world applications track statistical aggregates, in which extreme-value analysis provides useful insights about outliers.

Extreme-value analysis can also be extended to multivariate data with the use of distance- or depth-based methods [295, 343, 468]. However, these methods are applicable only to certain types of specialized scenarios in which outliers are known to be present at the boundaries of the data. Many forms of post-processing on multi-criteria outlier scores may use such methods. On the other hand, such methods are not very useful for *generic* outlier analysis, because of their inability to discover outliers in the sparse *interior* regions of a data set.

1.3.3 Probabilistic and Statistical Models

In probabilistic and statistical models, the data is modeled in the form of a closed-form probability distribution, and the parameters of this model are learned. Thus, the key assumption here is about the specific choice of the data distribution with which the modeling is performed. For example, a Gaussian mixture model assumes that the data is the output of a generative process in which each point belongs to one of k Gaussian clusters. The parameters of these Gaussian distributions are learned with the use of an *expectation-maximization (EM)* algorithm on the observed data so that the probability (or *likelihood*) of the process generating the data is as large as possible. A key output of this method is the membership probability of the data points to the different clusters, as well as the density-based fit to the modeled distribution. This provides a natural way to model the outliers, because data points that have very low fit values may be considered outliers. In practice, the logarithms of these fit values are used as the outlier scores because of the better propensity of the outliers to appear as extreme values with the use of log-fits. As discussed earlier, an extreme-value test may be applied to these fit values to identify the outliers.

A major advantage of probabilistic models is that they can be easily applied to virtually any data type (or mixed data type), as long as an appropriate generative model is available for each mixture component. For example, if the data is categorical, then a discrete Bernoulli distribution may be used to model each component of the mixture. For a mixture of different types of attributes, a product of the attribute-specific generative components may be used. Since such models work with probabilities, the issues of data normalization are already accounted for by the generative assumptions. Thus, probabilistic models provide a generic EM-based framework, which is relatively easy to apply to any specific data type. This is not necessarily the case with many other models.

A drawback of probabilistic models is that they try to fit the data to a particular kind of distribution, which may sometimes not be appropriate. Furthermore, as the number of model parameters increases, over-fitting becomes more common. In such cases, the outliers may fit the underlying model of normal data. Many parametric models are also harder to interpret in terms of intensional knowledge, especially when the parameters of the model cannot be intuitively presented to an analyst in terms of underlying attributes. This can defeat one of the important purposes of anomaly detection, which is to provide diagnostic understanding of the abnormal data generative process. A detailed discussion of probabilistic methods, including the EM algorithm, is provided in Chapter 2.

1.3.4 Linear Models

These methods model the data along lower-dimensional subspaces with the use of linear correlations [467]. For example, in the case of Figure 1.4, the data is aligned along a 1-dimensional line in a 2-dimensional space. The optimal line that passes through these points is determined with the use of regression analysis. Typically, a least-squares fit is used to determine the optimal lower-dimensional hyperplane. The distances of the data points from this hyperplane are used to quantify the outlier scores because they quantify the deviations from the model of normal data. Extreme-value analysis can be applied on these scores in order to determine the outliers. For example, in the 2-dimensional example of Figure 1.4, a linear model of the data points $\{(x_i, y_i), i \in \{1 \ldots N\}$ in terms of two coefficients a and b may be created as follows:

$$y_i = a \cdot x_i + b + \epsilon_i \quad \forall i \in \{1 \ldots N\} \tag{1.4}$$

Here, ϵ_i represents the *residual*, which is the modeling error. The coefficients a and b need to be *learned from the data* to minimize the least-squares error, which is denoted by $\sum_{i=1}^{N} \epsilon_i^2$. This is a convex non-linear programming problem whose solution can be obtained in closed form. The squared residuals provide the outlier scores. One can use extreme-value analysis to identify the unusually large deviations, which should be considered outliers.

The concept of dimensionality reduction and principal component analysis (PCA) is quite similar [296], except that it uses a non-parametric approach in order to model the data correlations. PCA can be derived through multivariate regression analysis by determining the hyperplane that minimizes the least-squares error (i.e., distance) to the hyperplane. In other words, it provides a subspace of lower dimensionality with the least *reconstruction error* after projection. Outliers have large reconstruction errors because they do not conform to the aggregate subspace patterns in the data. Therefore, the reconstruction errors may be used as outlier scores. In addition, principal component analysis can be used for noise *correction* [21], in which the attributes of data points are modified to reduce noise. Outlier

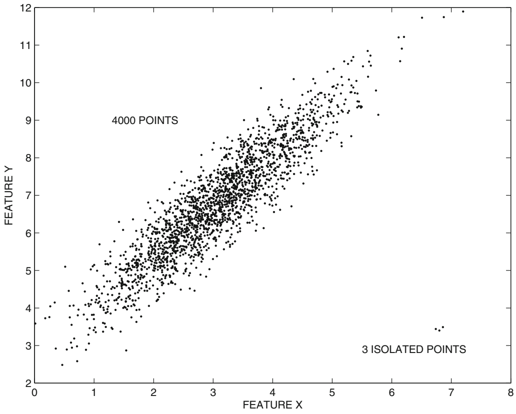


Figure 1.5: Small groups of anomalies can be a challenge to density-based methods

points are likely to be corrected more significantly than normal points. Methods like dimensionality reduction are special cases of the generic methodology of *matrix factorization*; the main advantage of the generic methodology is that it can even be used for incomplete data sets.

Dimensionality reduction and regression modeling are particularly hard to interpret in terms of the original attributes. This is because the subspace embedding is defined as a linear combination of attributes with positive or negative coefficients. This cannot easily be intuitively interpreted in terms specific properties of the data attributes. Nevertheless, some forms of dimensionality reduction, such as nonnegative matrix factorization, are highly interpretable. Dimensionality reduction, regression analysis, and matrix factorization methods for outlier detection are discussed in Chapter 3. Their natural nonlinear extensions such as kernel PCA, kernel SVMs, and neural networks, are also discussed in the same chapter. Furthermore, various forms of nonnegative matrix factorization are discussed in Chapters 8 and 12.

1.3.4.1 Spectral Models

Many of the matrix decomposition methods such as PCA are also used in the context of graphs and networks. The main difference is in how the matrix is created for decomposition. Some variations of these methods, which are used in certain types of data such as graphs and networks, are also referred to as spectral models. Spectral methods are used commonly for clustering graph data sets, and are often used in order to identify anomalous changes in temporal sequences of graphs [280]. Spectral methods are closely related to matrix factorization, which can also be used in such settings [551]. Such models will be discussed in Chapters 3, 4, 5, and 12.

1.3.5 Proximity-Based Models

The idea in proximity-based methods is to model outliers as points that are isolated from the remaining data on the basis of similarity or distance functions. Proximity-based methods are among the most popular class of methods used in outlier analysis. Proximity-based methods may be applied in one of three ways, which are clustering methods, density-based methods and nearest-neighbor methods. In clustering and other density-based methods, the dense regions in the data are found directly, and outliers are defined as those points that do not lie in these dense regions. Alternatively, one might define outliers as points that are located far away from the dense regions. The main difference between clustering and density-based methods is that clustering methods segment the data *points*, whereas the density-based methods such as histograms segment the data *space*. This is because the goal in the latter case is to estimate the density of test points in the data space, which is best achieved by space segmentation.

In nearest-neighbor methods [317, 456], the distance of each data point to its kth nearest neighbor is reported as its outlier score. By selecting a value of $k > 1$, small groups of tightly-knit points that are far away from the remaining data set can be identified and scored as outliers. It is reasonable to treat such sets of data points as outliers, because small related sets of points can often be generated by an anomalous process. For example, consider the case illustrated in Figure 1.5, which contains a large cluster containing 4000 data points, and a small set of isolated but three closely spaced and related anomalies. Such situations are quite common, because anomalies caused by the same (rare) process may result in small sets of data points that are almost identical. In this case, the points within an anomaly set are close to one another, and cannot be distinguished on the basis of the 1-nearest neighbor distance. Such anomalies are often hard to distinguish from noise by using certain types of density-based algorithms that are not sensitive to the global behavior of the data. On the other hand, the k-nearest neighbor approach can sometimes be effective. In the case of Figure 1.5, such sets of related anomalies may be identified by using $k \geq 3$. The kth nearest neighbor score provides an outlier score of the data set. This method can typically be computationally expensive, because it is required to determine the kth nearest neighbor of every point in the data set, which requires $O(N^2)$ operations for a data set containing N points. However, in cases in which it is acceptable to report binary labels instead of scores, many of these distance computations can be pruned because some points can be shown to be non-outliers after a small number of distance computations. For example, if after computing the distances of a small fraction of the points to a particular point 'A,' the k-nearest neighbor distance of 'A' is lower than that of all the top-r outliers found so far, then point 'A' is guaranteed to be a non-outlier. Therefore, no further distance computations to point 'A' need to be performed. As a result, the binary version of outlier detection often allows faster algorithms than the score-wise version of the problem. The latter is always quadratically related to the number of points in computational complexity. Quadratic computational complexity turns out to be surprisingly slow in real settings; it is often difficult to use these methods even for data sets containing a few hundred-thousand points, without leveraging some form of sampling.

In the case of clustering methods, the first step is to use a clustering algorithm in order to determine the dense regions of the data set. In the second step, some measure of the fit of the data points to the different clusters is used in order to compute an outlier score for the data point. For example, in the case of a k-means clustering algorithm, the distance of a data point to its nearest centroid may be used to measure its propensity to be an outlier. One needs to be careful with the use of clustering methods, because the specific data partitioning (and corresponding outlier scores) may vary significantly with the choice of clustering methodology. Therefore, it is usually advisable to cluster the data multiple times and average the scores obtained from the different runs [184, 406]. The results of such an approach are often surprisingly robust.

Density-based methods like histograms divide the data *space* into small regions, and the number of points in these regions are used to computed the outlier scores. Density-based methods provide a high level of interpretability, when the sparse regions in the data can be presented in terms of combinations of the original attributes. For example, consider a sparse region constructed on the following subsets of attributes:

$$\text{Age} \leq 20, \text{ Salary} \geq 100,000$$

Clearly, these constraints define a segment of the data space that is highly interpretable from a semantic point of view. It presents a clear description of *why* a data point should be considered an outlier. Some other methods such as kernel density estimation do not partition the data space, but are nevertheless focused on estimating the density of regions in the data space by replacing space segmentation with smoother kernel functions. Proximity-based methods for outlier detection are discussed in Chapter 4.

1.3.6 Information-Theoretic Models

Many of the aforementioned models for outlier analysis use various forms of data summarization such as generative probabilistic model parameters, clusters, or lower-dimensional hyperplanes of representation. These models implicitly generate a small *summary* of the data, and the *deviations* from this summary are flagged as outliers. Information-theoretic measures are also based on the same principle but in an indirect way. The idea is that outliers increase the minimum code length (i.e., minimum length of the *summary*) required to describe a data set because they represent *deviations* from natural attempts to summarize the data. For example, consider the following two strings:

```
ABABABABABABABABABABABABABABABABAB
ABABACABABABABABABABABABABABABABAB
```

The second string is of the same length as the first, and is different only at a single position containing the unique symbol ‘C.’ The first string can be described concisely as “AB 17 times.” However, the second string has a single position corresponding to the symbol ‘C.’ Therefore, the second string can no longer be described as concisely. In other words, the presence of the symbol ‘C’ somewhere in the string increases its minimum description length. It is also easy to see that this symbol corresponds to an outlier. Information-theoretic models are closely related to conventional models, because both use a concise representation of the data set as a baseline for comparison. For example, in the case of multidimensional data sets, both types of models use the following concise descriptions:

- A probabilistic model describes a data set in terms of generative model parameters, such as a mixture of Gaussian distributions or a mixture of exponential power distributions [92].

- A clustering or density-based summarization model describes a data set in terms of cluster descriptions, histograms, or other summarized representations, along with maximum error tolerances [284].

- A PCA model or spectral model describes the data in terms of lower dimensional subspaces of projection of multi-dimensional data or a condensed representation of a network [519], which is also referred to as its *latent representation.*

- A frequent pattern mining method describes the data in terms of an underlying code book of frequent patterns. These are among the most common methods used for information-theoretic anomaly detection [42, 151, 497].

All these models represent the data approximately in terms of individual condensed components representing aggregate trends. In general, outliers increase the length of the description in terms of these condensed components to achieve the same level of approximation. For example, a data set with outliers will require a larger number of mixture parameters, clusters, PCA-based subspace dimensionality, or frequent patterns in order to achieve *the same level of approximation.* Correspondingly, in information-theoretic methods, the key idea is to construct a *code book* in which to represent the data, and outliers are defined as points whose removal results in the *largest decrease* in description length [151], or the most accurate summary representation in the same description length after removal [284]. The term "code book" is rather loosely defined in outlier analysis and refers to the condensed aggregate components of the data in terms of which the data is described. The actual construction of the coding is often heuristic, and an effective choice is key to the success of the approach. In general, the determination of the minimum-length coding is a computationally intractable problem for a given data set, and therefore a variety of heuristic models (or code books) may be used for representation purposes [42, 151, 284, 497]. In many cases, these techniques can be related to conventional data summarization models for outlier analysis. In some cases, the coding is not explicitly constructed, but measures such as the entropy or Kolmogorov complexity are used as a surrogate in order to estimate the level of unevenness of a specific segment of the data [352, 312]. Segments with greater unevenness may be selectively explored to identify outliers. This represents a good use-case for information theoretic models because it is algorithmically simpler to quantify coding complexity than actually constructing the coding.

Conventional models look at this problem in a complementary way by *directly* defining outliers as points that are expressed in the *least precise way* by (or deviations from) from a *fixed* compression (e.g., clustering or factorization). On the other hand, information-theoretic models quantify the *differential impact on compression size* of removing an outlier point on a compression of fixed error (i.e., *aggregate* deviation). The two are clearly closely related, although the former is a more direct way of scoring points than the latter. Since information-theoretic methods largely differ from conventional models in terms of how the measure is defined, they often use similar methods as conventional techniques (e.g., probabilistic models [92], frequent pattern mining [42, 497], histograms [284], or PCA [519]) to create the coding representation. Therefore, most information-theoretic models cannot be considered a separate family from conventional models, and they will be discussed at various places in this book along with their conventional counterparts. It is noteworthy that the indirect approach used by information-theoretic models to score points can sometimes blunt the scores because of the impact of other points on the *aggregate* error. As a result, information-theoretic models often do not outperform their conventional counterparts. The best use-cases, therefore, arise is settings where quantifying the coding cost is algorithmically more convenient than directly measuring the deviations.

1.3.7 High-Dimensional Outlier Detection

The high-dimensional case is particularly challenging for outlier detection. The reason for this behavior is that many dimensions may be noisy and irrelevant for anomaly detection, which might also increase the propensity for pairwise distances to become more similar. The key point here is that *irrelevant* attributes have a dilution effect on the accuracy of distance computations and therefore the resulting outlier scores might also be inaccurate. When using distance-based algorithms to score outliers, one often observes the effect of weakly correlated and irrelevant attributes in the concentration of distances. In high-dimensional space, the data becomes increasingly sparse, and all pairs of data points become almost equidistant from one another [25, 263]. As a result, the outlier scores become less distinguishable from one another.

In such cases, outliers are best emphasized in a lower-dimensional *local* subspace of relevant attributes. This approach is referred to as *subspace outlier detection* [4], which is an important class of algorithms in the field of outlier analysis. The assumption in subspace outlier detection is that *outliers are often hidden in the unusual local behavior of low-dimensional subspaces, and this deviant behavior is masked by full-dimensional analysis.* Therefore, it may often be fruitful to explicitly search for the subspaces in which the anomalous behavior of points is best emphasized. This approach is a generalization of both (full-dimensional) clustering and (full-data) regression analysis. It combines local data pattern analysis with subspace analysis in order to mine the significant outliers. This can be a huge challenge, because the simultaneous discovery of relevant data localities and subspaces in high dimensionality can be computationally very difficult. It is easy to make mistakes in selecting the "correct" subspace, and it has been suggested [31, 35] that such techniques can be meaningfully used *only* by identifying multiple *relevant* subspaces and combining the predictions from these different subspaces. This approach is closely related to the notion of *outlier ensembles* [31, 35], which will be discussed in the next section and also in Chapter 6.

Subspace methods are useful for interpreting outliers, especially when the subspaces are described in terms of the original attributes. In such cases, the outputs of the algorithms provide *specific combinations of attributes along with data locality* that are relevant to the anomalous characteristics. This type of interpretability is useful in cases where a small number of *explanatory* attributes need to be identified from a high-dimensional data set. Methods for high-dimensional outlier detection are discussed in Chapter 5.

1.4 Outlier Ensembles

In many data mining problems such as clustering and classification, a variety of meta-algorithms are used in order to improve the robustness of the underlying solutions. Such meta-algorithms combine the outputs of multiple algorithms and are referred to as *ensembles.* For example, common ensemble methods in classification include bagging, subsampling, boosting and stacking [11, 33, 176]. Similarly, ensemble methods are often used to improve the quality of the clustering [23]. Therefore, it is natural to ask whether such meta-algorithms also exist for outlier detection. The answer is in the affirmative, although the work on meta-algorithms for outlier detection is relatively recent as compared to other problems like classification and clustering in which it is well-established. A position paper that formalizes these issues may be found in [31] and a book on outlier ensembles may be found in [35]. In recent years, significant theoretical and algorithmic advancements have been made in the field of outlier ensembles [32]. This chapter will provide a broad overview of the field of outlier ensembles, and a more detailed discussion is provided in Chapter 6. There are two primary types of ensembles in outlier analysis:

- In *sequential ensembles*, a given algorithm or set of algorithms are applied sequentially, so that future applications of the algorithms are influenced by previous applications, in terms of either modifications of the base data for analysis or in terms of the specific choices of the algorithms. The final result is either a weighted combination of, or the final result of the last application of an outlier analysis algorithm. For example, in the context of the classification problem, boosting methods may be considered examples of sequential ensembles.

```
Algorithm SequentialEnsemble(Data Set: $\mathcal{D}$
   Base Algorithms: $\mathcal{A}_1 \ldots \mathcal{A}_r$)
begin
 $j = 1$;
 repeat
  Pick an algorithm $\mathcal{A}_j$ based on results from
    past executions;
  Create a new data set $f_j(\mathcal{D})$ from $\mathcal{D}$ based
    on results from past executions;
  Apply $\mathcal{A}_j$ to $f_j(\mathcal{D})$;
  $j = j + 1$;
 until(termination);
 report outliers based on combinations of results
   from previous executions;
end
```

Figure 1.6: Sequential ensemble framework

- In *independent ensembles*, different algorithms, or different instantiations of the same algorithm are applied to either the complete data or portions of the data. The choices made about the data and algorithms applied are independent of the results obtained from these different algorithmic executions. The results from the different algorithm executions are combined together in order to obtain more robust outliers.

At a fundamental level, outlier ensembles are not very different from classification ensembles in terms of the underlying theoretical foundations [32]. Even though outlier detection is an unsupervised problem, the basic bias-variance theory from classification can be adapted to outlier detection by treating the underlying labels as unobserved [32]. As a result, many natural ensemble methods such as bagging and subsampling can be generalized easily to outlier detection with minor variations.

1.4.1 Sequential Ensembles

In sequential-ensembles, one or more outlier detection algorithms are applied sequentially to either all or portions of the data. The core principle of the approach is that each application of the algorithm enables a more refined execution with either a modified algorithm or data set. Thus, depending on the approach, either the data set or the algorithm may be changed in sequential executions. If desired, this approach can either be applied for a fixed number of times or executed to convergence. The broad framework of a sequential-ensemble algorithm is provided in Figure 1.6.

In each iteration, a successively refined algorithm is used on refined data based on results from previous executions. The function $f_j(\cdot)$ is used to create a refinement of the data, which could correspond to data subset selection, attribute-subset selection, or generic data transformation methods. The description above is provided in a very general form, and many special cases can be instantiated from this framework. For example, in practice, only a single algorithm may be used on successive modifications of the data, as data is refined over time. The sequential ensemble may be applied for a fixed number of iterations

```
**Algorithm** IndependentEnsemble(Data Set: $\mathcal{D}$
      Base Algorithms: $\mathcal{A}_1 \ldots \mathcal{A}_r$)
**begin**
  $j = 1$;
  **repeat**
    Pick an algorithm $\mathcal{A}_j$;
    Create a new data set $f_j(\mathcal{D})$ from $\mathcal{D}$;
    Apply $\mathcal{A}_j$ to $f_j(\mathcal{D})$;
    $j = j + 1$;
  **until**(termination);
  **report** outliers based on combinations of results
      from previous executions;
**end**
```

Figure 1.7: Independent ensemble framework

or to convergence. The broad principle of sequential ensembles is that a greater knowledge of data with successive algorithmic execution helps focus on techniques and portions of the data that can provide fresh insights.

Sequential ensembles have not been sufficiently explored in the outlier analysis literature as general purpose meta-algorithms. However, many *specific* techniques in the outlier literature use methods that can be recognized as special cases of sequential ensembles. A classic example of this is the use of two-phase algorithms for building a model of the normal data. In the first-phase, an outlier detection algorithm is used in order to remove the obvious outliers. In the second phase, *a more robust* normal model is constructed after removing these obvious outliers. Thus, the outlier analysis in the second stage is more accurate because many of the training points that contaminate the model of normal data have been removed. Such approaches are commonly used for cluster-based outlier analysis (for constructing more robust clusters in later stages) [70], or for more robust histogram construction and density estimation (see Chapter 4).

1.4.2 Independent Ensembles

In independent ensembles, different instantiations of the algorithm or different portions of the data are used for outlier analysis. Alternatively, the same algorithm may be applied with a different initialization, parameter set, or random seed. The results from these different algorithm executions can be combined in order to obtain a more robust outlier score. Such algorithms comprise the vast majority of outlier ensemble methods in use today. A general-purpose description of independent-ensemble algorithms is provided in the pseudo-code description of Figure 1.7.

The broad principle of independent ensembles is that different algorithms might perform better on different parts of the data; therefore, a combination of the results from these algorithms might provide more robust results than any of the individual ensemble components. The resulting output is therefore no longer as dependent on the specific artifacts of a particular algorithm or data set. Independent ensembles are used frequently for high-dimensional outlier detection, because they enable the exploration of different subspaces of the data in which different types of deviants may be found. In fact, the area of subspace outlier detection is deeply interconnected with outlier ensemble analysis (see Chapter 5).

There is a significant number of different ways in which different algorithms and training data sets may be leveraged for model combination. For example, the methods in [31, 32, 344, 367] sample subspaces from the underlying data in order to independently score outliers from each of these executions. Then, the scores from these different executions are unified into a single point-specific value. Similarly, methods such as bagging and subsampling, which combine the results from different training data sets in classification, have also been generalized to outlier detection [31, 32]. In some cases, randomized models are constructed by making randomized choices within an outlier scoring algorithm [368]. These methods will be discussed in Chapters 5 and 6.

1.5 The Basic Data Types for Analysis

Most of our aforementioned discussion is focused on multidimensional numerical data. Furthermore, it is assumed that the data records are independent of one another. However, in practice, the underlying data may be more complex both in attribute type and point-to-point dependence. Some examples of such real-world data types are discussed in this section.

1.5.1 Categorical, Text, and Mixed Attributes

Many data sets in real applications may contain categorical attributes that take on *discrete unordered* values. For example, demographic data might contain attributes such as race, gender, or ZIP code. Such attribute values are not ordered, and therefore require different analytical techniques. Mixed attribute data contain both numerical and categorical attributes. Most of the existing models can be extended to this case. In many cases, the major challenge is to construct a distance (or similarity) function that remains semantically meaningful for the case of discrete data.

Regression-based models can be used in a limited way over discrete attribute values, when the number of possible values of an attribute is not too large. The typical methodology is to convert the discrete data to binary data by creating one attribute for each categorical value. Regression models such as principal component analysis may then be applied to this binary data set. Such methods can be more easily extended to text, in which there is an inherent ordering among word frequencies. In such cases, the correlations among word occurrences can be used to create regression models. In fact, some of the most successful models for text de-noising are based on latent semantic analysis (LSA), which is a form of linear regression analysis [162]. Other common methods for text and categorical data include clustering [29], proximity-based methods [622], probabilistic models [578], and methods based on frequent pattern mining [42, 253, 497]. Methods for outlier detection in categorical, text, and mixed attribute data sets are discussed in Chapter 8.

1.5.2 When the Data Values have Dependencies

Most of the aforementioned discussion in this chapter is about the common multidimensional scenario, where it is assumed that the data records can be treated independently of one another. In practice, the different data values may be related to each other temporally, spatially, or through explicit network relationship links between the data items. The presence of such dependencies greatly changes the anomaly detection process even at the definition

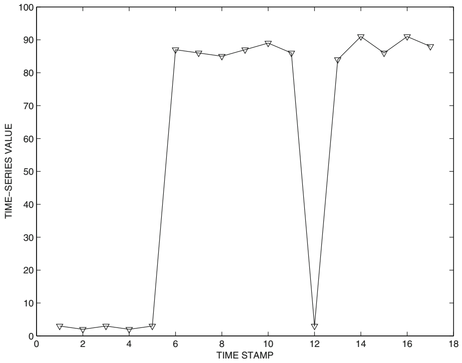


Figure 1.8: Example of Time Series

level. In such cases, the *expected values* of data items are influenced by their contextual dependencies, and therefore outliers are defined on the basis of such contextually modeled deviations. When a single data item (e.g., value from a time series) is declared as an anomaly because of its *relationship* to its related data items, it is referred to as a *contextual* outlier or anomaly. This is because such an outlier can only be understood in the *context* of its relationship with items in the temporal neighborhood. Such outliers are also sometimes referred to as *conditional anomalies* [503]. For example, a sudden spike in a time series is a contextual anomaly because it is very different from the values of its most recent time stamps; the key is to understand is that the most recent time stamps define the *expected* value of the series. Deviations from expected values represent outliers.

When a set of data items is declared anomalous *as a group of points*, it is referred to as a *collective* anomaly or outlier. For example, an unusual and rapid oscillation over time for a stock ticker value may be considered a collective anomaly, and it includes all the data items in the oscillation. Virtually, all anomalies in dependency-oriented data are *contextual* or *collective* anomalies, because they compute *expected* values based on relationships with adjacent data points in order to determine unexpected patterns. Furthermore, in such data sets, there are usually *multiple ways* to model anomalies, depending on what an analyst might be looking for. Some examples of such data domains are presented in this section.

1.5.2.1 Times-Series Data and Data Streams

Time series contains a set of values that are typically generated by continuous measurement over time. Therefore, the values in consecutive time stamps do not change very significantly, or change in a smooth way. In such cases, *sudden changes* in the underlying data records can be considered *anomalous events*. Therefore, the discovery of anomalous points in time series is closely related to the problem of anomalous *event detection*, and such events are often manifested as either *contextual or collective anomalies over related time stamps* [9, 19, 315]. The events are often created by sudden changes in the underlying system and may be of considerable interest to an analyst. For example, consider the following time series of values over consecutive time-stamps:

$$3, 2, 3, 2, 3, 87, 86, 85\ 87, 89, 86, 3, 84, 91, 86, 91, 88$$

The time series is illustrated in Figure 1.8. It is evident that there is a sudden change in the data value at time-stamp 6 from 3 to 87. This corresponds to an outlier. Subsequently, the data stabilizes at this value, *and this becomes the new normal.* At time-stamp 12, the data value again dips to 3. *Even though this data value was encountered before*, it is still considered an outlier because of the sudden change in the consecutive data values. Thus, it is critical to understand that in this case, treating the data values as independent of one another is not helpful for anomaly detection, because the data values are highly influenced by the adjacent values of the data points. In other words, temporal *context* is important. Thus, the problem of outlier detection in time-series data is highly related to the problem of change detection because the normal models of data values are highly governed by adjacency in temporal ordering. When completely new data values are encountered, they are referred to as *novelties* [391, 392, 388], although outlier detection is relevant to any form of abrupt change, rather than only new data values.

It should be emphasized that change analysis and outlier detection (in temporal data) are closely related problems, but they are not necessarily identical. The change in a temporal data set could occur in one of two possible ways:

- The values and trends in the data stream change slowly over time, a phenomenon that is referred to as *concept drift* [390, 10]. In such cases, the concept drift can only be detected by careful analysis over a longer period of time, and is not immediately obvious in many circumstances.

- The values and trends in the data stream change *abruptly*, so as to *immediately arouse suspicion that the underlying data generation mechanism has somehow changed fundamentally.*

Of the two scenarios, only the second one can be used to identify outliers. It is also easy to see the parallels between the second scenario and Hawkins's definition of outliers [249], which was introduced at the very beginning of this chapter.

A common challenge in such scenarios is to perform the outlier detection *in real time*, as new data values arrive. Many scenarios of change analysis and anomaly detection in temporal data are too tightly integrated to be treated separately. In such settings, solutions for one can be used for the other and vice versa. On the other hand, the modeling formulations of anomaly detection in temporal data are very diverse, not all of which are directly related to change detection. Usually, online analysis is suited to change detection, whereas offline analysis may explore other unusual aspects of the data. Some examples are as follows:

- When the data is in the form of a time series (e.g., sensor data) large changes in *trends* may correspond to anomalies. These can be discovered as deviations from forecasted values using window-based analysis. In some cases, it may be desired to determine time-series subsequences of unusual shapes rather than change points in the data.

- For multidimensional data streams, changes in the aggregate distribution of the streaming data may correspond to unusual events. For example, network intrusion events may cause *aggregate* change points in a network stream. On the other hand, *individual* point novelties may or may not correspond to aggregate change points. The latter case is similar to multidimensional anomaly detection with an efficiency constraint for the streaming scenario.

Methods for anomaly detection in time-series data and multidimensional data streams are discussed in Chapter 9.

1.5.2.2 Discrete Sequences

Many discrete sequence-based applications such as intrusion-detection and fraud-detection are clearly temporal in nature. This scenario can be considered a categorical or discrete analog of time-series data in which individual positions contain categorical (symbolic) values. Discrete sequences may not necessarily be temporal in nature, but may be based on their relative *placement* with respect to one another. An example is the case of biological data in which the sequences are defined by their relative placement.

As in the case of autoregressive models of continuous data, it is possible to use (typically Markovian) *prediction-based* techniques to forecast the value of a single *position* in the sequence. Deviations from forecasted values are identified as *contextual* outliers. It is often desirable to perform the prediction in real time in these settings. In other cases, anomalous events can be identified only by variations from the normal *patterns* exhibited by the *subsequences* over multiple time stamps. This is analogous to the problem of unusual *shape detection* in time-series data, and it represents a set of *collective* outliers.

Therefore, discrete sequences are analogous to continuous sequences, except that the categorical values in the individual positions necessitate the use of different similarity functions, representation data structures, and predictive techniques. For example, discrete sequence forecasting requires (more complex) Markovian models as opposed to (simpler) autoregressive techniques. The problem formulations in the two cases are, however, similar at a conceptual level. The specific techniques used are different because numerical time-series values are *ordered* and comparable across a continuous spectrum, whereas discrete values are not. As a result of these differences, the case of discrete sequences has been addressed in a different chapter from time-series data.

Discrete data are common in many real applications. Most biological sequences are discrete, and therefore the value of each position is drawn from a set of categorical possibilities. Similarly, host-based intrusion applications typically lead to discrete data, because numerous diagnostic events are drawn from a discrete set of instances [126]. Methods for anomaly detection in discrete sequences are discussed in Chapter 10.

1.5.2.3 Spatial Data

In spatial data, many non-spatial attributes (e.g., temperature, pressure, image pixel color intensity) are measured at spatial locations. Unusual local changes in such values are reported as outliers. It should be pointed out that outlier detection in temporal data shares some resemblance to that in spatial data [523]. Both typically require the attribute of interest to exhibit a certain level of continuity. For example, consider the measurement of the temperature in which the measurement could be associated with a time-stamp and spatial coordinates. Just as it is expected that temperatures at consecutive time-stamps do not vary too much (temporal continuity), it is also expected that temperatures at spatially close locations do not vary too much (spatial continuity). In fact, such unusual spatial variations in sea-surface temperatures and pressures [523] are used in order to identify significant and anomalous spatiotemporal events in the underlying data (e.g., formation of cyclones). Spatiotemporal data is a generalization of both spatial and temporal data, and the methods used in either domain can often be generalized to such scenarios. Methods for finding outliers in spatial and spatiotemporal data are discussed in Chapter 11.

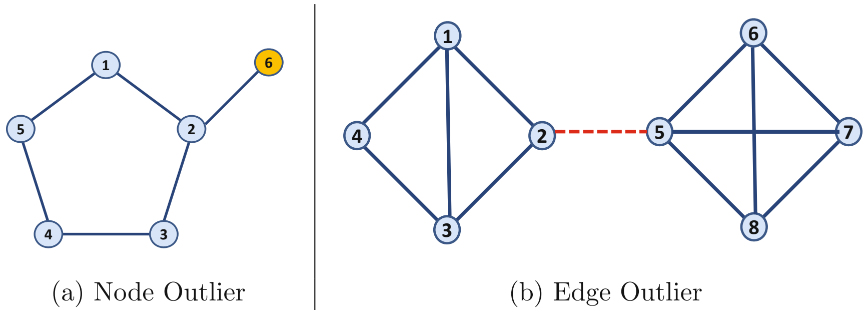


Figure 1.9: Examples of Node and Edge Outliers

1.5.2.4 Network and Graph Data

In network or graph data, the data values may correspond to nodes in the network, and the relationships among the data values may correspond to the edges in the network. In such cases, outliers may be modeled in different ways depending on the irregularity of *either* the nodes in terms of their relationships to other nodes, or the edges themselves. For example, a node that shows irregularity in its structure within its locality may be considered an outlier [41]. Similarly, an edge that connects disparate communities of nodes may be considered a *relationship* or *community outlier* [17, 214]. In Figure 1.9, two examples of outliers in networks are illustrated. Figure 1.9(a) illustrates an example of a node outlier because the node 6 has an unusual locality structure that is significantly different from other nodes. On the other hand, the edge $(2, 5)$ in Figure 1.9(b) may be considered a relationship outlier or community outlier, because it connects two distinct communities. Thus, there is greater complexity and flexibility in the definitions of outliers in complex data like graphs. There is also no unique way of defining the outliers and it is heavily dependent on the application domain at hand. In general, *the more complex the data is, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*

It is also possible to combine different types of dependencies for outlier modeling. For example, graphs may be temporal in nature. In such a case, the data may have both structural and temporal dependencies that change and influence each other over time [17]. Therefore, outliers may be defined in terms of significant changes in the underlying network community or distance structure. Such models combine network analysis and change detection to detect structural and temporal outliers. A detailed discussion of methods for temporal and non-temporal outlier detection in graphs is provided in Chapter 12. Relevant surveys are available in [14, 43, 457].

1.6 Supervised Outlier Detection

In many scenarios, previous examples of outliers may be available. A subset of the data may be labeled as anomalies, whereas the remaining data may be considered normal. In such cases, the anomaly identification process is referred to as *supervised outlier detection*, because labels are used in order to train a model that can determine *specific types* of anomalies. As a result, supervised models often provide *very different* results from the unsupervised case. For example, consider the following time-series:

$$3, 2, 3, 2, 3, 87, 2, 2, 3, 3, 3, 84, 91, 86, 91, 81$$

In this case, sudden changes in the data values (at 87 and 84) may be considered anomalies in the unsupervised scenario. However, in an application such as credit-card transaction levels, previous labeled examples of time-series may suggest that high consecutive values of the data should be considered anomalous. In such cases, the first occurrence of 87 should not be considered anomalous, whereas the occurrence of 84 along with its following values should be considered (collectively) anomalous.

As a general rule, one should always use supervision when labels are available because of its ability to discover application-specific anomalies of interest. Supervised outlier detection is a (difficult) special case of the classification problem. The main characteristic of this problem is that the labels are extremely unbalanced in terms of relative presence [132]. Since anomalies are far less common than normal points, it is possible for off-the-shelf classifiers to predict all test points as normal points and still achieve excellent accuracy. However, such results are not useful from a practical point of view. Therefore, the classifier is tuned, so that errors in classification of the anomalous class are penalized more heavily than the errors in classification of the majority class. The idea is that it is better to predict a negative class as an anomaly (false positive), rather than miss a true outlier (false negative). This leads to different trade-offs between false positives and false negatives than in other classification applications. These methods are referred to as *cost-sensitive learning*, because differential error costs are applied to different classes to regulate these trade-offs.

The supervised setting also supports several other variations of the classification problem that are quite challenging:

- A limited number of instances of the positive (outlier) class may be available, whereas the “normal” examples may contain an unknown proportion of outliers [183]. This is referred to as the Positive-Unlabeled Classification (PUC) problem in machine learning. This variation is still quite similar to the fully supervised rare-class scenario, except that the classification model needs to be more cognizant of the contaminants in the negative (unlabeled) class.

- Only instances of a subset of the normal and anomalous classes may be available, but some of the anomalous classes may be missing from the training data [388, 389, 538]. Such situations are quite common in scenarios such as intrusion detection in which some intrusions may be known, but other new types of intrusions are continually discovered over time. This is a *semi-supervised* setting for outlier detection. A combination of supervised and unsupervised methods may need to be used in such cases.

- In *active learning*, the problem of label acquisition is paired with the learning process [431]. The main assumption is that it is expensive to acquire examples of outliers, and therefore it is important to select the correct examples to label in order to perform accurate classification with the least number of labels.

Supervised methods for anomaly detection are discussed in Chapter 7.

1.7 Outlier Evaluation Techniques

A key question arises as to how the effectiveness of an outlier detection algorithm should be evaluated. Unfortunately, this is often a difficult task, because outliers, by definition, are rare. This means that the ground-truth labeling of data points as outliers or non-outliers is often not available. This is especially true for unsupervised algorithms, because if the ground-truth were indeed available, it could have been used to create a more effective supervised algorithm. In the unsupervised scenario (without ground-truth), it is often difficult to judge the effectiveness of the underlying algorithms in a rigorous way. Therefore, much of the research literature uses case studies to provide an intuitive and qualitative evaluation of the underlying outliers in unsupervised scenarios.

In other unsupervised problems like data clustering, a common approach is to use *internal validity measures*, in which a model of "goodness" is used to measure the effectiveness of the algorithm. For example, a common measure of goodness in data clustering is the mean-squared radius of a cluster. The main problem with such measures is that they only provide an idea of how well the model of "goodness" *matches* the model of learning. After all, there is no way of knowing the "correct" model of goodness in unsupervised problems; the paradox is that if we knew this correct model then we should use it in the algorithm rather than for evaluation. In fact, it is relatively easy to game such internal validity models by choosing an algorithm that is related to the model of goodness; this problem is well-known in the clustering domain [33]. This is also referred to as the problem of *over-fitting* in internal evaluation. In outlier detection, this problem is far more severe because a small number of changes in the labels of the outliers can drastically affect the performance. For example, a distance-based internal measure would favor a distance-based algorithm over a linear (e.g., PCA-based) technique. Conversely, a linear model of internal validity would favor a PCA-based technique over a distance-based algorithm. Therefore, internal validity measures are rarely used for outlier detection, which seems to be a wiser approach than has been adopted by the data-clustering community.

In outlier detection, a more reasonable (albeit imperfect) approach is to use *external validity measures*. In some cases, the data sets may be adapted from imbalanced classification problems, and the rare labels may be used as surrogates for the ground-truth outliers. In such cases, a natural question arises as to how the ground-truth can be used to evaluate effectiveness. Most outlier-detection algorithms output an outlier score, and a threshold on this score is used to convert the scores into outlier labels. If the threshold is selected too restrictively to minimize the number of declared outliers, then the algorithm will miss true outlier points (false negatives). On the other hand, if the algorithm declares too many data points as outliers, then it will lead to too many false positives. This trade-off can be measured in terms of *precision* and *recall*, which are commonly used for measuring the effectiveness of set-based retrieval.

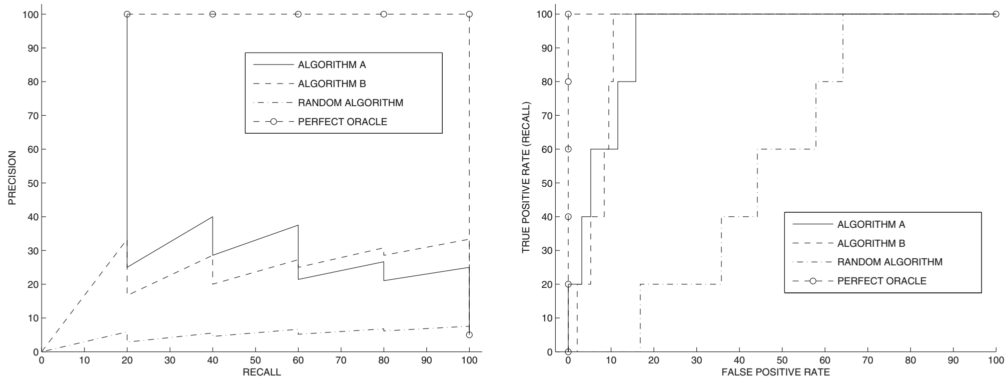
For any given threshold t on the outlier score, the declared outlier set is denoted by $S(t)$. As t changes, the size of $S(t)$ changes as well. G represent the true set (ground-truth set) of outliers in the data set. Then, for any given threshold t, the *precision* is defined as the percentage of *reported* outliers that truly turn out to be outliers.

$$Precision(t) = 100 \cdot \frac{|S(t) \cap G|}{|S(t)|} \tag{1.5}$$

The value of $Precision(t)$ is *not* necessarily monotonic in t, because both the numerator and denominator may change with t differently. The *recall* is correspondingly defined as the percentage of *ground-truth* outliers that have been reported as outliers at threshold t.

$$Recall(t) = 100 \cdot \frac{|S(t) \cap G|}{|G|} \tag{1.6}$$

By varying the parameter t, it is possible to plot a curve between the precision and the recall. This is referred to as the *Precision-Recall* curve. This curve is *not necessarily* monotonic.



(a) Precision-recall (b) Receiver operating characteristic

Figure 1.10: Precision-recall and receiver operating characteristic curves

Algorithm	Rank of Ground-truth Outliers
Algorithm A	1, 5, 8, 15, 20
Algorithm B	3, 7, 11, 13, 15
Random Algorithm	17, 36, 45, 59, 66
Perfect Oracle	1, 2, 3, 4, 5

Table 1.2: Rank of ground-truth outliers can be used to construct precision-recall curves

For more effective algorithms, high values of precision may often correspond to low values of recall and vice-versa. The precision-recall (PR) curve can also be generated by using thresholds on the *rank* of the data points, when sorted by outlier score. In the absence of ties in the outlier scores, a rank-based and score-based PR curve would be identical.

A *Receiver Operating Characteristic Curve (ROC)* is closely related to a Precision-Recall curve, but is sometimes visually more intuitive. In this case, the *True Positive Rate* is graphed against the *False Positive Rate*. The true positive rate $TPR(t)$ is defined in the same way as the recall. The false positive rate $FPR(t)$ is the percentage of the falsely reported positives out of the ground-truth negatives. In other words, the false-positive rate is a kind of "bad" recall, which reports the percentage of the negatives that are wrongly reported as outliers. Therefore, for a data set D with ground truth positives G, these definitions are as follows:

$$TPR(t) = Recall(t) = 100 \cdot \frac{|S(t) \cap G|}{|G|} \tag{1.7}$$

$$FPR(t) = BadRecall(t) = 100 \cdot \frac{|S(t) - G|}{|D - G|} \tag{1.8}$$

Therefore, the ROC curve plots the "bad" recall ($FPR(t)$) on the X-axis, and the "good" recall ($TPR(t)$) on the Y-axis. Note that both good and bad recall increase monotonically with the more relaxed values of the threshold t at which more outliers are reported. Therefore, the end points of the ROC curve are always at $(0, 0)$ and $(100, 100)$, and a random method is expected to exhibit performance along the diagonal line connecting these points. The *lift* obtained above this diagonal line provides an idea of the additional accuracy of the approach over a random method. The ROC curve is simply a different way to characterize the trade-offs than the precision-recall curve, although it has the advantage of being monotonic and more easily interpretable in terms of its lift characteristics.

In order to illustrate the insights gained from these different graphical representations, consider an example of a data set with 100 points, from which five points are outliers. Two algorithms A and B are applied to this data set, which rank all data points from 1 to 100, with a lower rank representing a greater propensity to be an outlier. Thus, the precision and recall values can be generated by determining the ranks of the 5 ground truth outlier points. In Table 1.2, some hypothetical ranks for the 5 ground truth outliers have been illustrated for the different algorithms. In addition, the ground truth ranks for a random algorithm have been indicated. This algorithm outputs a random outlier score for each point. Similarly, the ranks for a "perfect oracle" algorithm which ranks the correct top 5 points as outlier have also been illustrated in the table. The corresponding PR curve for this hypothetical output of outlier scores are illustrated in Figure 1.10(a). Other than the oracle algorithm, all the trade-off curves are non-monotonic. This is because the discovery of a new outlier at any particular relaxation in rank threshold results in a spike in the precision, which becomes less pronounced at higher values of the recall. The corresponding ROC curve is illustrated in Figure 1.10(b). Unlike the PR curve, this curve is clearly monotonic.

What do these curves really tell us? For cases in which one curve strictly dominates another the relative superiority between the two algorithms is unambiguous. For example, it is immediately evident that the oracle algorithm is superior to all algorithms, and the random algorithm is inferior to all the other algorithms. On the other hand, the algorithms A and B exhibit better performance at different parts of the ROC curve. In such cases, it is hard to say that one algorithm is strictly superior. From Table 1.2, it is clear that Algorithm A ranks three of the correct ground-truth outliers very highly, but the remaining two outliers are ranked poorly. In the case of Algorithm B, the highest ranked outliers are not as well ranked as the case of Algorithm A, although all five outliers are determined much earlier in terms of rank threshold. Correspondingly, Algorithm A dominates on the earlier part of the PR (or ROC) curve, whereas Algorithm B dominates on the later part. Some practitioners use the area under the ROC curve as a proxy for the overall effectiveness of the algorithm. A trapezoidal rule is used to compute the area, wherein the staircase-like ROC curve is replaced with a more convex approximation.

1.7.1 Interpreting the ROC AUC

The ROC AUC has the following simple probabilistic interpretation [246]:

Theorem 1.7.1 *Given a ranking or scoring of a set of points in order of their propensity to be outliers (with higher ranks/scores indicating greater outlierness), the ROC AUC is equal to the probability that a randomly selected outlier-inlier pair is ranked correctly (or scored in the correct order).*

In other words, one can also define the ROC AUC by computing the following mean over all outlier-inlier pairs in the data set:

$$\text{ROC AUC} = \text{MEAN}_{\overline{X_i} \in G, \overline{X_j} \in D-G} \begin{cases} 1 & \overline{X_i} \text{ ranked/scored higher than } \overline{X_j} \\ 0.5 & \overline{X_i} \text{ ranked/scored equal to } \overline{X_j} \\ 0 & \overline{X_i} \text{ ranked/scored lower than } \overline{X_j} \end{cases} \tag{1.9}$$

A nice characteristic of this definition is that it is easy to intuitively understand why a random algorithm would provide an AUC of around 0.5. This definition is also related to the Kendall rank-correlation coefficient, in which a similar computation is performed over *all* pairs of objects rather than only outlier-inlier pairs, and the range[2] of the reward function is drawn from $\{-1, 0, +1\}$ rather than $\{0, 0.5, 1\}$.

A measure, such as the AUC, should be used very carefully, because all parts of the ROC curve may not be equally important for different applications. This is because the initial part of the ROC curve is usually far more important. For example, for a data set containing 1000 points, it makes little difference whether an outlier data point is ranked at the $501th$ or $601th$ position. On the other hand, it makes a lot of difference whether an outlier data point is ranked at the $1st$ or $101th$ position. The AUC of the ROC makes little distinction between these two types of errors. In this sense, measures like precision can sometimes be more realistic in many settings. Other top-heavy measures such as the normalized discounted cumulative gain (NDCG) can also be adapted from the information retrieval and recommendation literature to outlier detection [34]. This measure computes the utility of a ranked list by given greater credit for outliers that are ranked at the top of the list.

1.7.2 Common Mistakes in Benchmarking

A common mistake made during the benchmarking of outlier detection algorithms occurs in cases where the algorithm is dependent on one or more user-defined parameters. For example, a k-nearest neighbor outlier detector scores a data point based on its k-nearest neighbor distance, where k is a user-defined parameter. It is common to run the algorithm repeatedly in order to select the best parameter at which the ROC AUC is optimized. Such an approach is not acceptable in outlier detection because one has effectively used knowledge of the outlier labels in selecting the parameters. In other words, the algorithm is no longer unsupervised. In problems like outlier detection, the only proper way to benchmark between a pair of algorithms is to run both over a "reasonable" range of parameters and compare the two algorithms using some central estimator of the performance over the resulting runs. For example, one can compare the median AUC performance or *box-plot performance*[3] of the two algorithms over the various parameter choices. Furthermore, different detectors might require the identification of a completely different range of parameters, which creates further challenges for comparison purposes. For example, a one-class support-vector machine might have a completely different choice of parameters than a nearest-neighbor detector, which further complicates a proper understanding of their relative performance. Unlike supervised settings, in which cross-validation is possible for selecting an approximately optimum parameter value for each classifier, the reasonable range for each outlier detector is often set on the basis of simple meta-characteristics of the data set such as its size and dimensionality. Clearly, these choices require some understanding and experience on the part of the analyst with various algorithms. There is only limited guidance available in the research literature on proper parameter choices for various algorithms.

A natural question arises as to whether one can use the *best* choice of parameters for *each* of the candidate algorithms to compare them. After all, such an approach does not *seem to* favor any particular algorithm over another since all algorithms are seeded with similar

[2] In the case of the Kendall rank-correlation coefficient, agreements are rewarded with +1, whereas disagreements are penalized with −1. Neutrals are credited values of 0. An outlier pair or an inlier pair would always belong to the neutral category. As a result, the absolute magnitude of the Kendall coefficient is more sensitive to the proportion of outliers in the data.

[3] See Figure 6.1 of Chapter 6 for an example. Box-plots are formally introduced in section 2.2.2.3 of Chapter 2.

knowledge. There are two problematic issues in doing so. First, the best choice of parameters cannot be known in unsupervised problems and therefore the results do not reflect the experience of an analyst in real-world settings. Second, such an evaluation will favor the more unstable algorithm, which is prone to overfitting into specific parameter choices. This will provide a skewed view of the algorithm to the analyst if the stable algorithm performs better than the unstable algorithm most of the time but the unstable algorithm performs extremely well at very specific parameter choices. After all, in an unsupervised setting, the likelihood of being correctly able to guess such parameter choices is similar to stumbling over a needle in a haystack. A second way of avoiding parametrization bias in comparing a pair of detectors [32] is to create an ensemble average of the execution of the algorithm over different settings, and compare the ensemble AUC of the different detectors. This truly provides an idea of the relative performance of the best possible avatars of various algorithms. At the end of the day, comparing two algorithms is somewhat of an art-form in the unsupervised setting, and one must rely at least a little on the experience and good judgement of the analyst in making proper experimental design choices.

1.8 Conclusions and Summary

The problem of outlier detection finds applications in numerous domains, where it is desirable to determine interesting and unusual events in the underlying generating process. The core of all outlier detection methods is the creation of a probabilistic, statistical or algorithmic model that characterizes the normal data. The deviations from this model are used to identify the outliers. A good domain-specific knowledge of the underlying data is often crucial designing simple and accurate models that do not overfit the underlying data. The problem of outlier detection becomes especially challenging, when significant relationships exist among the different data points. This is the case for time-series and network data in which the patterns in the relationships among the data points (whether temporal or structural) play the key role in defining the outliers. Outlier analysis has tremendous scope for further research, especially in the area of structural and temporal analysis.

1.9 Bibliographic Survey

A number of books and surveys have been written on the problem of outlier analysis. The classic books [74, 249, 467] in this area have mostly been written from the perspective of the statistics community. Most of these books were written before the wider adoption of database technology, and are therefore not written from a computational perspective. More recently, this problem has been studied quite extensively by the computer science community. These works consider practical aspects of outlier detection, corresponding to the cases where the data may be very large, or may have very high dimensionality. Numerous surveys have also been written that discuss the concept of outliers from different points of view, methodologies, or data types [38, 77, 125, 126, 313, 388, 389]. These surveys study outlier detection from different points of view such as the neural network setting [388, 389] or the one-class setting [313]. Among these, the survey by Chandola *et al.* [125] is the most recent and arguably the most comprehensive. This excellent review covers outlier detection quite broadly from the perspective of multiple communities. Detailed experimental comparisons of various outlier detection algorithms may be found in [35, 114, 184, 221, 419].

The basic models discussed in this chapter have also been researched extensively, and have been studied widely in the literature. Details of these methods (along with the corresponding bibliographic notes) will be provided in later chapters. Here, only the most important works in each area are covered. The key statistical techniques on regression-based modeling are covered in [467]. The Z-value test discussed in section 1.2 is used commonly in the statistical literature, and many variants for limited sample sizes such as the Grubb's test [225] and t-value test are also available. The basic EM-algorithm for unsupervised modeling of data sets was first proposed in [164] and leveraged for outlier detection in [578]. The non-parametric technique of principal component analysis (PCA) discussed in section 1.2 is described well in [296]. The core technique of PCA was extended to text (with some minor variations) as Latent Semantic Indexing [162]. A variety of distance-based methods for outlier detection are proposed in [317, 456, 533], and density-based methods for outlier detection were proposed in [96]. Methods for interpreting distance-based outliers were first proposed in [318]. A variety of information-theoretic methods for outlier detection are discussed in [42, 57, 92, 122, 151, 256, 257, 352, 497].

The issues of poor behavior of high-dimensional applications (such as clustering and nearest-neighbor search) have been observed in several prior works in the literature [5, 7, 8, 25, 263]. The problem of high-dimensional outlier detection was first proposed in [4]. Subspace approaches for outlier detection were proposed in this paper, and a number of other recent methods have followed a similar line of work [308, 327, 402, 403, 404, 406, 604, 605, 606, 607, 619].

Outliers have been studied extensively in the context of different data domains. While numeric data is the most commonly studied case, numerous methods have also been proposed for categorical and mixed data [38, 578]. Methods for unsupervised outlier detection in text corpora are proposed in [240]. The problem of detecting outliers with dependencies has also been studied extensively in the literature. Methods for detecting outliers and changes in time series and streams were proposed in [9, 17, 19, 29, 310, 311, 312, 315]. Novelty detection [388] is an area that is closely related to outlier analysis, and it is often studied in the context of supervised models, where novel classes from a data stream are detected in real time [391, 392] with the use of learning methods. However, novelty detection is also studied often in the unsupervised scenario, particularly in the context of *first story detection* in topic detection and tracking in text streams [622]. Spatial outliers [2, 324, 376, 487, 488, 489, 490] are closely related to the problem of finding outliers in temporal data, since such data also show spatial continuity, just as temporal data show temporal continuity. Some forms of spatial data also have a temporal component to them, which requires the determination of spatiotemporal outliers [141, 142]. Outlier detection in discrete sequences is related to the problem of temporal outlier detection in continuous sequences. For discrete sequences, an excellent survey may be found in [126]. General surveys on anomaly detection in various types of temporal data may be found in [231, 232].

Methods for finding node outliers with unusual neighborhood behavior in graphs were proposed in [41], and techniques for finding relationship outliers, subgraph outliers and community outliers were proposed in [17, 214, 416, 452]. The primary ideas in all these methods is that outlier regions in a network are caused by unusual relationships in the form of edges, subgraphs, and communities. The problem of evolutionary network analysis in temporal networks is studied in [20, 233, 234, 519]. Surveys on anomaly detection in static and dynamic networks may be found in [14, 43, 457].

Recently, methods for *outlier ensembles* have been proposed. The work in [344] designs methods for using different subsets of features in outlier detection methods, and combining them in order to provide more effective results. The work in [402, 403, 404] shows how to combine the scores from different subspaces found by outlier detection algorithms in order to provide a unified and more robust result. The work in [367] proposes the notion of *isolation forests* that are analogs of the successful notion of random forests in classification. Recently, the field has been formalized by positioning the existing (informal) work in the field of outlier ensembles, and also establishing theoretical foundations [31, 32, 35].

The supervised version of the outlier detection problem has been studied extensively in the form of *rare class detection*. For the supervised case, readers are referred to a general book on classification [176], since this problem is essentially a cost-sensitive variation [132, 182] on the standard classification problem, in which the class distributions are very imbalanced. In particular, the readers are referred to [132, 182] for a thorough discussion on the foundations of cost-sensitive learning from imbalanced data sets. A number of methods for classification from positive and unlabeled data are discussed in [183], and a good review of the previous work in this area may also be found from the references in this paper. The work in [431, 618, 619] first showed how human supervision could be used to significantly improve the effectiveness of outlier detection. Finally, the *semi-supervised* scenario of novelty detection has been discussed extensively in [388, 389, 538].

Evaluation methods in outlier analysis are identical to the techniques used in information retrieval, recommender systems, and (supervised) rare-class learning. In fact, most of the evaluation methods discussed in the Chapter 7 of a recent recommender systems book [34] can also be used for outlier analysis. A detailed discussion of ROC curves may be found in [192]. While the ROC and PR curves are the traditional methods for outlier evaluation, it has recently been noted [402] that these methods may not necessarily provide all the insights needed for different types of analysis. Therefore, the work in [402] has proposed a coefficient based on the Spearman correlation between the best possible ranking and the ranking determined by the algorithm.

1.10 Exercises

1. Which of the following points from each of the following sets of points below is an outlier? Why?

- (1-dimensional) { 1, 3, 2, 1, 3, 2, 75, 1, 3, 2, 2, 1, 2, 3, 2, 1 }
- (1-dimensional) { 1, 2, 3, 4, 2, 19, 9, 21, 20, 22 }
- (2-dimensional) { (1, 9), (2, 9), (3, 9), (10, 10), (10, 3), (9, 1), (10, 2) }

2. Use MATLAB or any other mathematical software to create a histogram of the data distribution along each of the dimensions in the different cases of Exercise 1. Can you see the outliers visually? Which ones? In which case are the outliers not clear and why?

3. For the 2-dimensional case of Exercise 1, plot the data points on a 2-dimensional plane. Can you see the outliers visually? Which ones?

4. Apply the Z-value test to each of the cases in Exercise 1. For the 2-dimensional case, apply the Z-value test to the individual dimensions. Do you discover the correct outliers?

5. For the 2-dimensional case in Exercise 1, construct the function $f(x_1, x_2) = |x_1 - x_2|$. Apply the Z-value test to $f(x_1, x_2)$ over each of the data points. Do you obtain the correct outliers, as suggested by your visual analysis in Exercise 3? Why?

6. Determine the nearest neighbor of each data point for the cases in Exercise 1. Which data points have the largest value of the nearest neighbor distance? Are they the correct outliers?

7. Apply a k-means clustering algorithm to each of the cases in Exercise 1, while setting $k = 2$. Which data points lie furthest from the two means thus found? Are these the correct outliers?

8 Consider the following time-series:

- 1, 2, 3, 3, 2, 1, 73, 1, 2, 3, 5
- 1, 2, 3, 4, 3, 2, 1, 3, 73, 72, 74, 73, 74, 1, 2, 3, 4, 2
- 1, 2, 3, 5, 6, 19, 11, 15, 17, 2, 17, 19 , 17, 18

Which data points would you consider outliers? How does the temporal component influence your choice of outliers? Now examine the points at which the time series changes significantly? How do these points relate to the outliers?

9. Consider the undirected network $G = (N, A)$ of 8 nodes in N indexed from 1 through 8. Let the edge set A be { (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8) }. Draw the network on paper to visualize it. Is there any node, which you would consider an outlier? Why?

- Now delete the edge $(1, 7)$. Does this change the set of nodes you would consider outliers? Why?

10. Consider the undirected network $G = (N, A)$ of 8 nodes in N indexed from 1 through 8. Let the edge set A be { (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (5, 7), (4, 7), (5, 6), (6, 8), (5, 8), (6, 7) }. Draw the network on paper to visualize it. Is there any edge, which you would consider an outlier? Why?

11. Consider three algorithms A, B and C, which are run on a data set with 100 points and 5 outliers. The rank of the outliers by score for the three algorithms are as follows:
A: 1, 3, 5, 8, 11
B: 2, 5, 6, 7, 9
C: 2, 4, 6, 10, 13
Draw the PR curves for each of the algorithms. Would you consider any of the algorithms strictly superior to any of the others? Why?