CHAPTER

11 Outlier detection

Imagine that you are a transaction auditor in a credit card company. To protect your customers from credit card fraud, you pay special attention to card usages that are rather different from typical cases. For example, if a purchase amount is much bigger than usual for a card owner, and if the purchase occurs far from the owner's resident city, then the purchase is suspicious. You want to detect such transactions as soon as they occur and contact the card owner for verification. This is common practice in many credit card companies. *What data mining techniques can help detect suspicious transactions?*

Most credit card transactions are normal. However, if a credit card is stolen, its transaction pattern usually changes dramatically—the locations of purchases and the items purchased are often very different from those of the authentic card owner and other customers. An essential idea behind credit card fraud detection is to identify those transactions that are very different from the norm.

Outlier detection (also known as *anomaly detection*) is the process of finding data objects with behaviors that are very different from expectation. Such objects are called **outliers** or **anomalies**. Outlier or anomaly detection is important in many applications in addition to fraud detection such as medical care, public safety and security, industry damage detection, image processing, sensor and video network surveillance, national security, and intrusion detection.

Outlier detection and clustering analysis are two highly related tasks. Clustering finds the majority patterns in a data set and organizes the data accordingly, whereas outlier detection tries to capture those exceptional cases that deviate substantially from the majority patterns. Outlier detection and clustering analysis serve different purposes. In terms of methodologies, outlier detection might also use supervision during the detection process, whereas clustering analysis is typically unsupervised in nature.

In this chapter, we study outlier detection techniques. Section 11.1 introduces the basic concepts, including different types of outliers and an overview of outlier detection methods. In the rest of the chapter, you will learn about outlier detection methods in detail. These approaches, organized here by category, are statistical (Section 11.2), proximity-based (Section 11.3), reconstruction-based (Section 11.4), and clustering-based vs. classification-based approaches (Section 11.5). In addition, you will learn about mining contextual and collective outliers (Section 11.6) and outlier detection in high-dimensional data (Section 11.7).

11.1 Basic concepts

Let us first define what outliers are, categorize the different types of outliers, and then discuss the challenges in outlier detection at a general level, followed by an overview of outlier detection methods.

Data Mining. https://doi.org/10.1016/B978-0-12-811760-6.00021-7

11.1.1 What are outliers?

Assume that a given statistical process is used to generate a set of data objects. An **outlier** is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism. For ease of presentation within this chapter, we may refer to data objects that are not outliers as "normal" or expected data. Similarly, we may refer to outliers as "abnormal" data.

Example 11.1. Outliers. In Fig. 11.1, most objects roughly follow a Gaussian distribution. However, the objects in region R are significantly different. It is unlikely that they follow the same distribution as the other objects in the data set. Thus, the objects in R are outliers in the data set. □

Outliers are different from noisy data. As mentioned in Chapter 2, noise is a random error or variance in a measured variable. In general, noise is not interesting in data analysis, including outlier detection. For example, in credit card fraud detection, a customer's purchase behavior can be modeled as a random variable. A customer may generate some "noisy transactions" that may seem like "random errors" or "variance," such as by buying a bigger lunch one day, or having one more cup of coffee than usual. Such transactions should not be treated as outliers; otherwise, the credit card company would incur heavy costs from verifying that many transactions. The company may also lose customers by bothering them with multiple false alarms. As in many other data analysis and data mining tasks, noise should be removed before outlier detection.

Outliers are interesting because they are suspected of not being generated by the same mechanism as the rest of the data. Therefore in outlier detection, it is important to justify *why* the detected outliers are generated by some other mechanisms. This is often achieved by making various assumptions on the rest of the data and showing that the detected outliers violate those assumptions significantly.

Outlier detection is also related to *novelty detection* in evolving data sets. For example, by monitoring a social media web site where new content is incoming, novelty detection may identify new topics and trends in a timely manner. Novel topics may initially appear as outliers. To this extent, outlier detection and novelty detection share some similarity in modeling and detection methods. However, a critical difference between the two is that in novelty detection, once new topics are confirmed, they are usually incorporated into the model of normal behavior so that follow-up instances are not treated as outliers anymore.

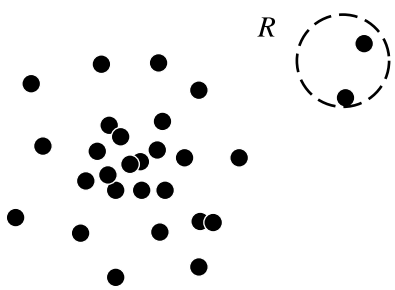


FIGURE 11.1

The objects in region R are outliers.

11.1.2 Types of outliers

In general, outliers can be classified into three categories, namely global outliers, contextual (or conditional) outliers, and collective outliers. Let's examine each of these categories.

Global outliers

In a given data set, a data object is a **global outlier** if it deviates significantly from the rest of the data set. Global outliers are sometimes called *point anomalies* and are the simplest type of outliers. Most outlier detection methods are aimed at finding global outliers.

Example 11.2. Global outliers. Consider the points in Fig. 11.1 again. The points in region R significantly deviate from the rest of the data set and hence are examples of global outliers. □

To detect global outliers, a critical issue is to find an appropriate measurement of deviation with respect to the application in question. Various measurements exist, and, based on these, outlier detection methods are partitioned into different categories. We will come to this issue in detail later.

Global outlier detection is important in many applications. Consider intrusion detection in computer networks, for example. If the communication behavior of a computer is very different from the normal patterns (e.g., a large number of packages is broadcast in a short time), this behavior may be considered as a global outlier, and the corresponding computer is a suspected victim of hacking. As another example, in trading transaction auditing systems, transactions that do not follow the regulations are considered as global outliers and should be held for further examination.

Contextual outliers

"The temperature today is 28 °*C. Is it exceptional (**i.e.**, an outlier)?"* It depends, for example, on the time and location! If it is in winter in Toronto, yes, it is an outlier. If it is a summer day in Toronto, then it is normal. Unlike global outlier detection, in this case, whether or not today's temperature value is an outlier depends on the context—the date, the location, and possibly some other factors.

In a given data set, a data object is a **contextual outlier** if it deviates significantly with respect to a specific context of the object. Contextual outliers are also known as *conditional outliers* because they are conditional on the selected context. Therefore, in contextual outlier detection, the context has to be specified as part of the problem definition. Generally, in contextual outlier detection, the attributes of the data objects in question are divided into two groups:

- **Contextual attributes**: The contextual attributes of a data object define the object's context. In the temperature example, the contextual attributes may be date and location.
- **Behavioral attributes**: These define the object's characteristics and are used to evaluate whether the object is an outlier in the context to which it belongs. In the temperature example, the behavioral attributes may be the temperature, humidity, and pressure.

Unlike global outlier detection, in contextual outlier detection, whether a data object is an outlier depends on not only the behavioral attributes but also the contextual attributes. A configuration of behavioral attribute values may be considered an outlier in one context (e.g., 28 °C is an outlier for a Toronto winter) but not an outlier in another context (e.g., 28 °C is not an outlier for a Toronto summer).

Contextual outliers are a generalization of local outliers, a notion introduced in density-based outlier analysis approaches. An object in a data set is a **local outlier** if its density significantly deviates from the local area in which it occurs. We will discuss local outlier analysis in greater detail in Section 11.3.2.

Global outlier detection can be regarded as a special case of contextual outlier detection where the set of contextual attributes is empty. In other words, global outlier detection uses the whole data set as the context. Contextual outlier analysis provides flexibility to users in that one can examine outliers in different contexts, which can be highly desirable in many applications.

Example 11.3. Contextual outliers. In credit card fraud detection, in addition to global outliers, an analyst may consider outliers in different contexts. Consider customers who use more than 90% of their credit limit. If one such customer is viewed as belonging to a group of customers with low credit limits, then such behavior may not be considered an outlier. However, similar behavior of customers from a high-income group may be considered outliers if their balance often exceeds their credit limit. Such outliers may lead to business opportunities—raising credit limits for such customers can bring in new revenue. □

The quality of contextual outlier detection in an application depends on the meaningfulness of the contextual attributes, in addition to the measurement of the deviation of an object to the majority in the space of behavioral attributes. More often than not, the contextual attributes should be determined by domain experts, which can be regarded as part of the input background knowledge. In many applications, neither obtaining sufficient information to determine contextual attributes nor collecting high-quality contextual attribute data is easy.

"How can we formulate meaningful contexts in contextual outlier detection?" A straightforward method simply uses group-bys of the contextual attributes as contexts. This may not be effective, however, because many group-bys may have insufficient data or noise. A more general method uses the proximity of data objects in the space of contextual attributes. We discuss this approach in detail in Section 11.3.

Collective outliers

Suppose you are a supply-chain manager of an electronics store. You handle thousands of orders and shipments every day. If the shipment of an order is delayed, it may not be considered an outlier because, statistically, delays occur from time to time. However, you have to pay attention if 100 orders are delayed on a single day. Those 100 orders as a whole form an outlier, although each of them may not be regarded as an outlier if considered individually. You may have to take a close look at those orders collectively to understand the shipment problem.

Given a data set, a subset of data objects forms a **collective outlier** if the objects as a whole deviate significantly from the entire data set. Importantly, the individual data objects may not be outliers.

Example 11.4. Collective outliers. In Fig. 11.2, the black objects as a whole form a collective outlier because the density of those objects is much higher than the rest in the data set. However, every black object individually is not an outlier with respect to the whole data set. □

Collective outlier detection has many important applications. For example, in intrusion detection, a denial-of-service package from one computer to another is considered normal and not an outlier at all. However, if several computers keep sending denial-of-service packages to each other, they as a whole should be considered as a collective outlier. The computers involved may be suspected of being compromised by an attack. As another example, a stock transaction between two parties is considered normal. However, a large set of transactions of the same stock among a small party in a short period are collective outliers because they may be evidence of some people manipulating the market.

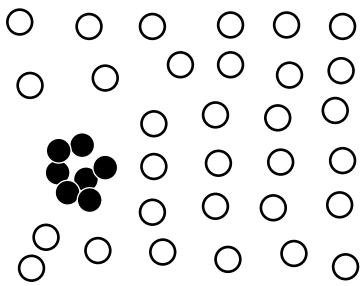


FIGURE 11.2

The black objects form a collective outlier.

Unlike global or contextual outlier detection, in collective outlier detection we have to consider not only the behavior of individual objects, but also that of groups of objects. Therefore to detect collective outliers, we need background knowledge of the relationship among data objects such as distance or similarity measurements between objects.

In summary, a data set can have multiple types of outliers. Moreover, an object may belong to more than one type of outlier. In business, different outliers may be used in various applications or for different purposes. Global outlier detection is the simplest. Contextual outlier detection requires background information to determine contextual attributes and contexts. Collective outlier detection requires background information to model the relationship among objects to find groups of outliers.

11.1.3 Challenges of outlier detection

Outlier detection is useful in many applications yet faces many challenges such as the following:

- **Modeling normal objects and outliers effectively.** Outlier detection quality highly depends on the modeling of normal (nonoutlier) objects and outliers. Often, building a comprehensive model for data normality is very challenging, if not impossible. This is partly because it is hard to enumerate all possible normal behaviors in an application.
  The border between data normality and abnormality (outliers) is often not clear-cut. Instead, there can be a wide range of gray area. Consequently, while some outlier detection methods assign to each object in the input data set a label of either "normal" or "outlier," other methods assign to each object a score measuring the "outlier-ness."[1]
- **Application-specific outlier detection.** Technically, choosing the similarity or distance measure and the relationship model to describe data objects is critical in outlier detection. Unfortunately, such choices are often application-dependent. Different applications may have very different requirements. For example, in clinic data analysis, a small deviation may be important enough to justify an outlier. In contrast, in marketing analysis, objects are often subjected to larger fluctuations, and consequently a substantially larger deviation is needed to justify an outlier. Outlier detection's

[1] In some literature, it is also referred to as the "outlyingness" score.

high dependency on the application type makes it impossible to develop a universally applicable outlier detection method. Instead, individual outlier detection methods that are dedicated to specific applications must be developed.

- **Handling noise in outlier detection.** As mentioned earlier, outliers are different from noise. It is also well known that the quality of real data sets tends to be poor. Noise often unavoidably exists in data collected in many applications. Noise may be present as deviations in attribute values or even as missing values. Low data quality and the presence of noise bring a huge challenge to outlier detection. They can distort the data, blurring the distinction between normal objects and outliers. Moreover, noise and missing data may "hide" outliers and reduce the effectiveness of outlier detection—an outlier may appear "disguised" as a noisy point, and an outlier detection method may mistakenly identify a noisy point as an outlier.
- **Interpretability.** In some application scenarios, a user may want to not only detect outliers but also understand why the detected objects are outliers. To meet the understandability requirement, an outlier detection method has to provide some justification of the detection. For example, a statistical method can be used to justify the degree to which an object may be an outlier based on the likelihood that the object was generated by the same mechanism that generated the majority of the data. The smaller the likelihood, the more unlikely the object was generated by the same mechanism, and the more likely the object is an outlier.

11.1.4 An overview of outlier detection methods

There are many outlier detection methods in the literature and in practice. Here, we present two orthogonal ways to categorize outlier detection methods. First, we categorize outlier detection methods according to whether the sample of data for analysis is given with domain expert–provided labels that can be used to build an outlier detection model. Second, we divide methods into groups according to their assumptions regarding normal objects vs. outliers.

Supervised, semisupervised, and unsupervised methods

If expert-labeled examples of normal or outlier objects can be obtained, they can be used to build outlier detection models. The methods used can be divided into supervised methods, semisupervised methods, and unsupervised methods.

Supervised methods. Supervised methods model data normality and abnormality. Domain experts examine and label a sample of the underlying data. Outlier detection can then be modeled as a classification problem (Chapters 6 and 7). The task is to learn a classifier that can recognize outliers. The sample is used for training and testing. In some applications, the experts may label just the normal objects, and any other objects not matching the model of normal objects are reported as outliers. Other methods model the outliers and treat objects not matching the model of outliers as normal.

Although many classification methods can be applied, challenges to supervised outlier detection include the following:

- The two classes (i.e., normal objects vs. outliers) are imbalanced. That is, the population of outliers is typically much smaller than that of normal objects. Therefore methods for handling imbalanced classes (Section 6.7.5) may be used, such as oversampling (i.e., replicating) outliers to increase their distribution in the training set used to construct the classifier. Due to the small population of

outliers in data, the sample data examined by domain experts and used in training may not even sufficiently represent the outlier distribution. The lack of outlier samples can limit the capability of the constructed classifiers. To tackle these problems, some methods "make up" artificial outliers.

- In many outlier detection applications, catching as many outliers as possible (i.e., the sensitivity or recall of outlier detection) is far more important than not mislabeling normal objects as outliers. Consequently, when a classification method is used for supervised outlier detection, it has to be interpreted appropriately so as to consider the application interest on recall.

In summary, supervised methods of outlier detection must be careful in how they train and how they interpret classification rates due to the fact that outliers are rare in comparison to the other data samples.

Unsupervised methods. In some application scenarios, objects labeled as "normal" or "outlier" are not available. Thus an unsupervised learning method has to be used.

Unsupervised outlier detection methods make an implicit assumption: The normal objects are somewhat "clustered." In other words, an unsupervised outlier detection method expects that normal objects follow a pattern far more frequently than outliers. Normal objects do not have to fall into one group sharing high similarity. Instead, they can form multiple groups, where each group has distinct features. However, an outlier is typically expected to occur far away in feature space from any of those groups of normal objects.

This assumption may not be true all the time. For example, in Fig. 11.2, the normal objects do not share any strong patterns. Instead, they are uniformly distributed. The collective outliers, however, share high similarity in a small area. Unsupervised methods might not be able to detect such outliers effectively. In some applications, normal objects are diversely distributed, and many such objects do not follow strong patterns. For instance, in some intrusion detection and computer virus detection problems, normal activities are very diverse and many do not fall into high-quality clusters. In such scenarios, unsupervised methods may have a high false positive rate—they may mislabel many normal objects as outliers (intrusions or viruses in these applications), and let many actual outliers go undetected. Due to the high similarity between intrusions and viruses (i.e., they have to attack key resources in the target systems), modeling outliers using supervised methods may be far more effective.

Many clustering methods can be adapted to act as unsupervised outlier detection methods. The central idea is to find clusters first, and then the data objects not belonging to any cluster could be flagged as outliers. However, such methods suffer from two issues. First, a data object not belonging to any cluster may be noise instead of an outlier. Second, it is often costly to find clusters first and then find outliers. It is usually assumed that there are far fewer outliers than normal objects. Having to process a large population of nontarget data entries (i.e., the normal objects) before one can touch the real meat (i.e., the outliers) can be unappealing. More recent unsupervised outlier detection methods develop various smart ideas to tackle outliers directly without explicitly and completely finding clusters. You will learn more about these techniques in Sections 11.3 and 11.5.1 on proximity-based and clustering-based methods, respectively.

Semisupervised methods. In many applications, although obtaining some labeled examples is feasible, the number of such labeled examples is often small. We may encounter cases where only a small set of the normal and outlier objects are labeled, but most of the data are unlabeled. Semisupervised outlier detection methods were developed to tackle such scenarios.

Semisupervised outlier detection methods can be regarded as applications of semisupervised learning methods (Section 7.5.1). For example, when some labeled normal objects are available, we can use

them, together with unlabeled objects that are close by, to train a model for normal objects. The model of normal objects then can be used to detect outliers—those objects not fitting the model of normal objects are classified as outliers.

If only some labeled outliers are available, semisupervised outlier detection is trickier. A small number of labeled outliers are unlikely to represent all the possible outliers. Therefore building a model for outliers based on only a few labeled outliers is unlikely to be effective. To improve the quality of outlier detection, we can get help from models for normal objects learned from unsupervised methods.

For additional information on semisupervised methods, interested readers are referred to the bibliographic notes at the end of this chapter.

Statistical methods, proximity-based methods, and reconstruction-based methods

As discussed earlier, outlier detection methods make assumptions about outliers vs. the rest of the data. According to the assumptions made, we can categorize outlier detection methods into three types: statistical methods, proximity-based methods, and reconstruction-based methods.

Statistical methods. **Statistical methods** (also known as **model-based methods**) make assumptions of data normality. They assume that normal data objects are generated by a statistical (stochastic) model, and that data not following the model are outliers.

Example 11.5. Detecting outliers using a statistical (Gaussian) model. In Fig. 11.1, the data points except for those in region R fit a Gaussian distribution g_D, where for a location $\boldsymbol{x}$ in the data space, $g_D(\boldsymbol{x})$ gives the probability density at $\boldsymbol{x}$. Thus, the Gaussian distribution g_D can be used to model the normal data, that is, most of the data points in the data set. For each object $\boldsymbol{y}$ in region R, we can estimate $g_D(\boldsymbol{y})$, the probability that this point fits the Gaussian distribution. Because $g_D(\boldsymbol{y})$ is very low, $\boldsymbol{y}$ is unlikely generated by the Gaussian model and thus is an outlier. □

The effectiveness of statistical methods highly depends on whether the assumptions made for the statistical model hold true for the given data. There are many kinds of statistical models. For example, the statistic models used in the methods may be parametric or nonparametric. Statistical methods for outlier detection are discussed in detail in Section 11.2.

Proximity-based methods. **Proximity-based methods** assume that an object is an outlier if the nearest neighbors of the object are far away in feature space, that is, the proximity of the object to its neighbors significantly deviates from the proximity of most of the other objects to their neighbors in the same data set.

Example 11.6. Detecting outliers using proximity. Consider the objects in Fig. 11.1 again. If we model the proximity of an object using its three nearest neighbors, then the objects in region R are substantially different from other objects in the data set. For the two objects in R, their second and third nearest neighbors are dramatically more remote than those of any other objects. Therefore we can label the objects in R as outliers based on proximity. □

The effectiveness of proximity-based methods relies heavily on the proximity (or distance) measure used. In some applications, such measures cannot be easily obtained. Moreover, proximity-based methods often have difficulty in detecting a group of outliers if the outliers are close to one another. There are two major types of proximity-based outlier detection, namely *distance-based* and *density-based* outlier detection. Proximity-based outlier detection is discussed in Section 11.3.

Reconstruction-based methods. Reconstruction-based outlier detection approaches are built upon the following idea. Since the normal data samples often share certain similarities, they can often be represented in a more succinct way compared with their original representation (e.g., an attribute vector for each data sample). With the succinct representation, we can well *reconstruct* the original representation of the normal samples. On the other hand, for samples that cannot be well reconstructed by such alternative, succinct representation, we flag them as outliers.

There are two major types of reconstruction-based outlier detection approaches, namely matrix-factorization based methods for numerical data; and pattern-based compression methods for categorical data. Reconstruction-based outlier detection is discussed in detail in Section 11.4.

The rest of this chapter discusses approaches to outlier detection.

11.2 Statistical approaches

As with statistical methods for clustering, statistical methods for outlier detection make assumptions about data normality. They assume that the normal objects in a data set are generated by a stochastic process (e.g., a generative model). Consequently, normal objects occur in the regions of high probability for the stochastic model, and objects in the regions of low probability are outliers.

The general idea behind statistical methods for outlier detection is to learn a generative model fitting the given data set and then identify those objects in low-probability regions of the model as outliers. However, there are many different ways to learn generative models. In general, statistical methods for outlier detection can be divided into two major categories: *parametric methods* and *nonparametric methods*, according to how the models are specified and learned.

A **parametric method** assumes that the normal data objects are generated by a parametric distribution with a finite number of parameters Θ. The *probability density function* of the parametric distribution $f(\boldsymbol{x}, \Theta)$ gives the probability that object $\boldsymbol{x}$ is generated by the distribution. The smaller this value, the more likely $\boldsymbol{x}$ is an outlier.

A **nonparametric method** does not assume an a priori statistical model with a finite number of parameters. Instead, a nonparametric method tries to determine the model from the input data. Note that most nonparametric methods do not assume that the model is completely parameter-free. (Such an assumption would make learning the model from data almost a mission impossible.) Instead, nonparametric methods often take the position that the number and nature of the parameters are flexible and not fixed in advance. Examples of nonparametric methods include histogram and kernel density estimation.

11.2.1 Parametric methods

In this subsection, we introduce several simple yet practical parametric methods for outlier detection. We first discuss methods for univariate data based on normal distribution. We then discuss how to handle multivariate data using multiple parametric distributions.

Detection of univariate outliers based on normal distribution

Data involving only one attribute or variable are called *univariate data*. For simplicity, we often choose to assume that data are generated from a normal distribution. We can then learn the parameters of the normal (i.e., Gaussian) distribution from the input data and identify the points with low probability as outliers.

Let's start with univariate data. We will try to detect outliers by assuming the data follow a normal distribution.

Example 11.7. Univariate outlier detection using maximum likelihood method. Suppose a city's average temperature values in July in the last 10 years are, in value-ascending order, 24.0 °C, 28.9 °C, 28.9 °C, 29.0 °C, 29.1 °C, 29.1 °C, 29.2 °C, 29.2 °C, 29.3 °C, and 29.4 °C. Let's assume that the average temperature follows a normal distribution, which is determined by two parameters: the mean, μ, and the standard deviation, σ.

We can use the *maximum likelihood method* to estimate the parameters μ and σ. That is, we maximize the *log-likelihood function*

$$\ln \mathcal{L}(\mu, \sigma^2) = \sum_{i=1}^{n} \ln f(x_i | (\mu, \sigma^2)) = -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2, \tag{11.1}$$

where n is the total number of samples, which is 10 in this example.

Taking derivatives with respect to μ and σ^2 and solving the resulting system of first-order conditions leads to the following *maximum likelihood estimates*:

$$\hat{\mu} = \overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{11.2}$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2. \tag{11.3}$$

In this example, we have

$$\begin{aligned} \hat{\mu} &= \frac{24.0 + 28.9 + 28.9 + 29.0 + 29.1 + 29.1 + 29.2 + 29.2 + 29.3 + 29.4}{10} = 28.61 \\ \hat{\sigma}^2 &= ((24.0 - 28.61)^2 + (28.9 - 28.61)^2 + (28.9 - 28.61)^2 + (29.0 - 28.61)^2 \\ &\quad + (29.1 - 28.61)^2 + (29.1 - 28.61)^2 + (29.2 - 28.61)^2 + (29.2 - 28.61)^2 \\ &\quad + (29.3 - 28.61)^2 + (29.4 - 28.61)^2)/10 \backsimeq 2.29. \end{aligned}$$

Accordingly, we have $\hat{\sigma} = \sqrt{2.29} = 1.51$.

The most deviating value, 24.0 °C, is 4.61 °C away from the estimated mean. We know that the $\mu \pm 3\sigma$ region contains 99.7% data under the assumption of normal distribution. Because $\frac{4.61}{1.51} = 3.04 > 3$, the probability that the value 24.0 °C is generated by the normal distribution is less than 0.15% and thus can be identified as an outlier. □

Example 11.7 elaborates a simple yet practical outlier detection method. It simply labels any object as an outlier if it is more than 3σ away from the mean of the estimated distribution, where σ is the standard deviation.

Such straightforward methods for statistical outlier detection can also be used in visualization. For example, the *boxplot method* (described in Chapter 2) plots the univariate input data using a five-number summary (Fig. 11.3): the smallest nonoutlier value (Min), the lower quartile ($Q1$), the median ($Q2$), the

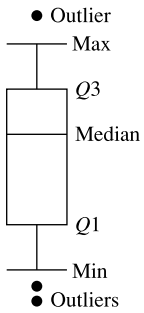


FIGURE 11.3

Using a boxplot to visualize outliers.

upper quartile ($Q3$), and the largest nonoutlier value (Max). The *interquantile range* (*IQR*) is defined as $Q3 - Q1$. Any object that is more than $1.5 \times IQR$ smaller than $Q1$ or $1.5 \times IQR$ larger than $Q3$ is treated as an outlier because the region between $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$ contains 99.3% of the objects. The rationale is similar to using 3σ as the threshold for normal distribution.

Another simple statistical method for univariate outlier detection using normal distribution is the *Grubb's test* (also known as the *maximum normed residual test*). For each object x in a data set, we define a z-score as

$$z = \frac{|x - \mu|}{\sigma}, \tag{11.4}$$

where μ is the mean, and σ is the standard deviation of the input data. An object x is an outlier if

$$z \geq \frac{n-1}{\sqrt{n}} \sqrt{\frac{t^2_{\alpha/(2n),n-2}}{n-2+t^2_{\alpha/(2n),n-2}}}, \tag{11.5}$$

where $t^2_{\alpha/(2n),n-2}$ is the value taken by a t-distribution at a significance level of $\alpha/(2n)$, and n is the number of objects in the data set.

Detection of multivariate outliers

Data involving two or more attributes or variables are *multivariate data*. Many univariate outlier detection methods can be extended to handle multivariate data. The central idea is to transform the multivariate outlier detection task into a univariate outlier detection problem. Here, we use two examples to illustrate this idea.

Example 11.8. Multivariate outlier detection using the Mahalanobis distance. For a multivariate data set, let $\bar{\boldsymbol{o}}$ be the sample mean vector. For an object $\boldsymbol{o}$ in the data set, the squared Mahalanobis distance from $\boldsymbol{o}$ to $\bar{\boldsymbol{o}}$ is

$$MDist(\boldsymbol{o}, \bar{\boldsymbol{o}}) = (\boldsymbol{o} - \bar{\boldsymbol{o}})^T S^{-1} (\boldsymbol{o} - \bar{\boldsymbol{o}}), \tag{11.6}$$

where S is the sample covariance matrix.

$MDist(\boldsymbol{o}, \bar{\boldsymbol{o}})$ is a univariate variable, and thus Grubb's test can be applied to this measure. Therefore we can transform the multivariate outlier detection tasks as follows:

1. Calculate the mean vector from the multivariate data set.
2. For each object $\boldsymbol{o}$, calculate $MDist(\boldsymbol{o}, \bar{\boldsymbol{o}})$, the squared Mahalanobis distance from $\boldsymbol{o}$ to $\bar{\boldsymbol{o}}$.
3. Detect outliers in the transformed univariate data set, $\{MDist(\boldsymbol{o}, \bar{\boldsymbol{o}}) | \boldsymbol{o} \in D\}$.
4. If $MDist(\boldsymbol{o}, \bar{\boldsymbol{o}})$ is determined to be an outlier, then $\boldsymbol{o}$ is regarded as an outlier as well. □

Our second example uses the χ^2-statistic to measure the distance between an object to the mean of the input data set.

Example 11.9. Multivariate outlier detection using the χ^2-statistic. The χ^2-statistic can also be used to capture multivariate outliers under the assumption of normal distribution. For an object, $\boldsymbol{o}$, the χ^2-statistic is

$$\chi^2 = \sum_{i=1}^{n} \frac{(\boldsymbol{o}_i - E_i)^2}{E_i}, \tag{11.7}$$

where $\boldsymbol{o}_i$ is the value of $\boldsymbol{o}$ on the ith dimension, E_i is the mean of the i-dimension among all objects, and n is the dimensionality. If the χ^2-statistic is large, the object is an outlier. □

Using a mixture of parametric distributions

If we assume that the data are generated by a normal distribution, this works well in many situations. However, this assumption may be overly simplified when the actual data distribution is complex. In such cases, we instead assume that the data are generated by a mixture of parametric distributions.

Example 11.10. Multivariate outlier detection using multiple parametric distributions. Consider the data set in Fig. 11.4. There are two big clusters, C_1 and C_2. To assume that the data are generated by a normal distribution would not work well here. The estimated mean is located between the two clusters and not inside any cluster. The objects between the two clusters cannot be detected as outliers since they are close to the mean. □

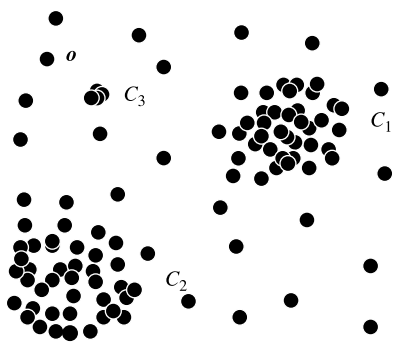


FIGURE 11.4

A complex data set.

To overcome this problem, we can instead assume that the normal data objects are generated by multiple normal distributions, two in this case. That is, we assume two normal distributions, $\Theta_1(\mu_1, \sigma_1)$ and $\Theta_2(\mu_2, \sigma_2)$. For any object $\boldsymbol{o}$ in the data set, the probability that $\boldsymbol{o}$ is generated by the mixture of the two distributions is given by

$$Pr(\boldsymbol{o}|\Theta_1, \Theta_2) = w_1 f_{\Theta_1}(\boldsymbol{o}) + w_2 f_{\Theta_2}(\boldsymbol{o}),$$

where f_{Θ_1} and f_{Θ_2} are the probability density functions of Θ_1 and Θ_2, respectively, and w_1 and w_2 are the weights of two probability density functions. We can use the *expectation-maximization* (EM) algorithm (Chapter 9) to learn the parameters μ_1, σ_1, μ_2, σ_2, w_1, and w_2 from the data, as we do in mixture models for clustering. Each cluster is represented by a learned normal distribution. An object $\boldsymbol{o}$ is detected as an outlier if it does not belong to any cluster, that is, the probability is very low that it was generated by the combination of the two distributions.

Example 11.11. Multivariate outlier detection using multiple clusters. Most of the data objects shown in Fig. 11.4 are in either C_1 or C_2. Other objects, representing noise, are uniformly distributed in the data space. A small cluster, C_3, is highly suspicious because it is not close to either of the two major clusters, C_1 and C_2. The objects in C_3 should therefore be detected as outliers.

Note that identifying the objects in C_3 as outliers is difficult, whether or not we assume that the given data follow a normal distribution or a mixture of multiple distributions. This is because the probability of the objects in C_3 will be higher than some of the noise objects, like $\boldsymbol{o}$ in Fig. 11.4, due to a higher local density in C_3. □

To tackle the problem demonstrated in Example 11.11, we can assume that the normal data objects are generated by a normal distribution, or a mixture of normal distributions, whereas the outliers are generated by another distribution. Heuristically, we can add constraints on the distribution that is generating outliers. For example, it is reasonable to assume that this distribution has a larger variance if the outliers are distributed in a larger area. Technically, we can assign $\sigma_{outlier} = k\sigma$, where k is a user-specified parameter and σ is the standard deviation of the normal distribution generating the normal data. Again, the EM algorithm can be used to learn the parameters.

11.2.2 Nonparametric methods

In nonparametric methods for outlier detection, the model of "normal data" is learned from the input data, rather than assuming one a priori. Nonparametric methods often make fewer assumptions about the data, and thus can be applicable in more scenarios.

Example 11.12. Outlier detection using a histogram. An electronics store records the purchase amount for every customer transaction. Fig. 11.5 uses a histogram (refer to Chapter 2) to graph these amounts as percentages, given all transactions. For example, 60% of the transaction amounts are between \$0.00 and \$1000.

We can use the histogram as a nonparametric statistical model to capture outliers. For example, a transaction in the amount of \$7500 can be regarded as an outlier because only $1 - (60\% + 20\% + 10\% + 6.7\% + 3.1\%) = 0.2\%$ of transactions have an amount higher than \$5000. On the other hand, a transaction amount of \$385 can be treated as normal because it falls into the bin (or bucket) holding 60% of the transactions. □

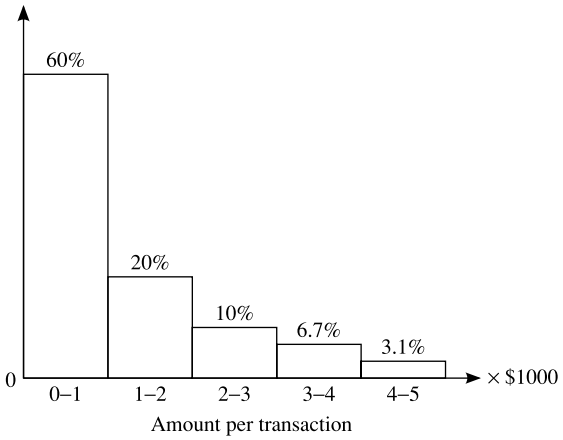


FIGURE 11.5

Histogram of purchase amounts in transactions.

As illustrated in the previous example, the histogram is a frequently used nonparametric statistical model that can be used to detect outliers. The procedure involves the following two steps.

Step 1: Histogram construction. In this step, we construct a histogram using the input data (training data). The histogram may be univariate as in Example 11.12 or multivariate if the input data are multidimensional.

Note that although nonparametric methods do not assume a priori statistical model, they often do require user-specified parameters to learn models from data. For example, to construct a good histogram, a user has to specify the type of histogram (e.g., equal width or equal depth) and other parameters (e.g., the number of bins in the histogram or the size of each bin). Unlike parametric methods, these parameters do not specify types of data distribution (e.g., Gaussian).

Step 2: Outlier detection. To determine whether an object $\boldsymbol{o}$ is an outlier, we can check it against the histogram. In the simplest approach, if the object falls in one of the histogram's bins, the object is regarded as normal. Otherwise, it is considered an outlier.

For a more sophisticated approach, we can use the histogram to assign an outlier-ness score to the object. In Example 11.12, we can let an object's outlier-ness score be the reciprocal of the volume of the bin in which the object falls. For example, the outlier-ness score for a transaction amount of \$7500 is $\frac{1}{0.2\%} = 500$ and that for a transaction amount of \$385 is $\frac{1}{60\%} = 1.67$. The scores indicate that the transaction amount of \$7500 is much more likely to be an outlier than that of \$385.

A drawback of using histograms as a nonparametric model for outlier detection is that it is hard to choose an appropriate bin size. On the one hand, if the bin size is set too small, many normal objects may end up in empty or rare bins and thus be misidentified as outliers. This leads to a high false positive rate and low precision. On the other hand, if the bin size is set too high, outlier objects may infiltrate into some frequent bins and thus be "disguised" as normal. This leads to a high false negative rate and low recall.

To overcome this problem, we can adopt kernel density estimation to estimate the probability density distribution of the data. We treat an observed object as an indicator of high probability density in the surrounding region. The probability density at a point depends on the distances from this point to the observed objects. We use a *kernel function* to model the influence of a sample point within its neighborhood. A kernel $K()$ is a nonnegative real-valued integrable function that satisfies the following two conditions:

- $\int_{-\infty}^{+\infty} K(u)du = 1$.
- $K(-u) = K(u)$ for all values of u.

A frequently used kernel is a standard Gaussian function with mean 0 and variance 1:

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}. \tag{11.8}$$

Let $x_1, \ldots, x_n$ be an independent and identically distributed samples of a random variable f. The kernel density approximation of the probability density function is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{11.9}$$

where $K()$ is a kernel and h is the bandwidth serving as a smoothing parameter.

Once the probability density function of a data set is approximated through kernel density estimation, we can use the estimated density function $\hat{f}$ to detect outliers. For an object $\boldsymbol{o}$, $\hat{f}(o)$ gives the estimated probability that the object is generated by the stochastic process. If $\hat{f}(o)$ is high, then the object is likely normal. Otherwise, o is likely an outlier. This step is often similar to the corresponding step in parametric methods.

In summary, statistical methods for outlier detection learn models from data to distinguish outliers from normal data objects. An advantage of using statistical methods is that the outlier detection may be statistically justifiable. Of course, this is true only if the statistical assumption made about the underlying data meets the constraints in reality.

The data distribution of high-dimensional data is often complicated and hard to be fully understood. Consequently, statistical methods for outlier detection on high-dimensional data remain a big challenge. Outlier detection for high-dimensional data is further addressed in Section 11.7.

The computational cost of statistical methods depends on the models. When simple parametric models are used (e.g., a Gaussian), fitting the parameters typically takes linear time. When more sophisticated models are used (e.g., mixture models, where the EM algorithm is used in learning), approximating the best parameter values often takes several iterations. Each iteration, however, is typically linear with respect to the data set's size. For kernel density estimation, the model learning cost can be up to quadratic. Once the model is learned, the outlier detection cost is often very small per object.

11.3 Proximity-based approaches

Given a set of objects in feature space, a distance measure can be used to quantify the similarity between objects. Intuitively, objects that are far from others can be regarded as outliers. Proximity-based approaches assume that the proximity of an outlier object to its nearest neighbors significantly deviates from the proximity of most other objects to their nearest neighbors in the data set.

There are two types of proximity-based outlier detection methods: distance-based and density-based methods. A *distance-based outlier detection method* consults the **neighborhood** of an object, which is defined by a given radius. An object is then considered as an outlier if its neighborhood does not have enough other points. A *density-based outlier detection method* investigates the density of an object and that of its neighbors. Here, an object is identified as an outlier if its density is relatively much lower than that of its neighbors.

Let's start with distance-based outliers.

11.3.1 Distance-based outlier detection

A representative method of proximity-based outlier detection uses the concept of **distance-based outliers**. For a set, D, of data objects to be analyzed, a user can specify a distance threshold, r, to define a reasonable neighborhood of an object. For each object, $\boldsymbol{o}$, we can examine the number of other objects in the r-neighborhood of $\boldsymbol{o}$. If most of the objects in D are far from $\boldsymbol{o}$, that is, not in the r-neighborhood of $\boldsymbol{o}$, then $\boldsymbol{o}$ can be regarded as an outlier.

Formally, let r $(r \geq 0)$ be a *distance threshold* and π $(0 < \pi \leq 1)$ be a fraction threshold. An object, $\boldsymbol{o}$, is a $DB(r, \pi)$**-outlier** if

$$\frac{\|\{\boldsymbol{o}' | dist(\boldsymbol{o}, \boldsymbol{o}') \leq r\}\|}{\|D\|} \leq \pi, \tag{11.10}$$

where $dist(\cdot, \cdot)$ is a distance measure.

Equivalently, we can determine whether an object, $\boldsymbol{o}$, is a $DB(r, \pi)$-outlier by checking the distance between $\boldsymbol{o}$ and its k-nearest neighbor, $\boldsymbol{o}_k$, where $k = \lceil \pi \|D\| \rceil$. Object $\boldsymbol{o}$ is an outlier if $dist(\boldsymbol{o}, \boldsymbol{o}_k) > r$, because in such a case, there are fewer than k objects except for $\boldsymbol{o}$ that are in the r-neighborhood of $\boldsymbol{o}$.

"How can we compute $DB(r, \pi)$-outliers?" A straightforward approach is to use nested loops to check the r-neighborhood for every object, as shown in Fig. 11.6. For any object, $\boldsymbol{o}_i$ $(1 \leq i \leq n)$, we calculate the distance between $\boldsymbol{o}_i$ and the other object and count the number of other objects in the r-neighborhood of $\boldsymbol{o}_i$. Once we find $\pi \cdot n$ other objects within a distance r from $\boldsymbol{o}_i$, the inner loop can be terminated because $\boldsymbol{o}_i$ already violates (Eq. (11.10)), and thus is not a $DB(r, \pi)$-outlier. On the other hand, if the inner loop completes for $\boldsymbol{o}_i$, this means that $\boldsymbol{o}_i$ has less than $\pi \cdot n$ neighbors in a radius of r, and thus is a $DB(r, \pi)$-outlier.

The straightforward nested loop approach takes $O(n^2)$ time. Surprisingly, the actual CPU runtime is often linear with respect to the data set size. For most nonoutlier (i.e., normal) objects, the inner loop terminates early when the number of outliers in the data set is small, which should be the case most of the time. Correspondingly, only a small fraction of the data set is examined.

Algorithm: Distance-based outlier detection.

Input:

- a set of objects $D = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n\}$, threshold r $(r > 0)$ and π $(0 < \pi \leq 1)$;

Output: $DB(r, \pi)$ outliers in D.
Method:

```
for i = 1 to n do
    count ← 0
    for j = 1 to n do
        if i ≠ j and dist(o_i, o_j) ≤ r then
            count ← count + 1
            if count ≥ π · n then
                exit {o_i cannot be a DB(r, π) outlier}
            endif
        endif
    endfor
    print o_i {o_i is a DB(r, π) outlier according to (Eq. 11.10)}
endfor;
```

FIGURE 11.6

Nested loop algorithm for $DB(r, \pi)$-outlier detection.

11.3.2 Density-based outlier detection

Distance-based outliers, such as $DB(r, \pi)$-outliers, are just one type of outlier. Specifically, distance-based outlier detection takes a global view of the data set. Such outliers can be regarded as "global outliers" for two reasons:

- A $DB(r, \pi)$-outlier, for example, is far (as quantified by parameter r) from at least $(1 - \pi) \times 100\%$ of the objects in the data set. In other words, an outlier as such is remote from the majority of the data.
- To detect distance-based outliers, we need two global parameters, r and π, which are applied to every outlier object.

Many real-world data sets demonstrate a more complex structure, where objects may be considered outliers with respect to their local neighborhoods rather than with respect to the global data distribution. Let's look at an example.

Example 11.13. Local proximity-based outliers. Consider the data points in Fig. 11.7. There are two clusters: C_1 is dense, and C_2 is sparse. Object $\boldsymbol{o}_3$ can be detected as a distance-based outlier because it is far from the majority of the data set.

Now, let's consider objects $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$. Are they outliers? On the one hand, the distance from $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ to the objects in the dense cluster, C_1, is smaller than the average distance between an object in cluster C_2 and its nearest neighbor. Thus, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are not distance-based outliers. In fact, if we were to categorize $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ as $DB(r, \pi)$-outliers, we would have to classify all the objects in cluster C_2 as $DB(r, \pi)$-outliers.

On the other hand, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ can be identified as outliers when they are considered locally with respect to cluster C_1 because $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ deviate significantly from the objects in C_1. Moreover, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are also far from the objects in C_2.

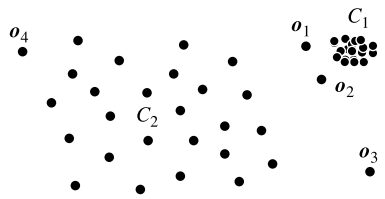


FIGURE 11.7

Global outliers and local outliers.

To summarize, distance-based outlier detection methods cannot capture local outliers like $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$. Note that the distance between object $\boldsymbol{o}_4$ and its nearest neighbors is much greater than the distance between $\boldsymbol{o}_1$ and its nearest neighbors. However, because $\boldsymbol{o}_4$ is local to cluster C_2 (which is sparse), $\boldsymbol{o}_4$ is not considered a local outlier. □

"How can we formulate the local outliers as illustrated in Example 11.13?" The critical idea here is that we need to compare the density around an object with the density around its local neighbors. The basic assumption of density-based outlier detection methods is that the density around a nonoutlier object is similar to the density around its neighbors, while the density around an outlier object is significantly different from the density around its neighbors.

Based on the preceding, density-based outlier detection methods use the relative density of an object against its neighbors to indicate the degree to which an object is an outlier.

Now, let's consider how to measure the *relative density* of an object, $\boldsymbol{o}$, given a set of objects, D. The *k-distance* of $\boldsymbol{o}$, denoted by $dist_k(\boldsymbol{o})$, is the distance, $dist(\boldsymbol{o}, \boldsymbol{p})$, between $\boldsymbol{o}$ and another object, $\boldsymbol{p} \in D$, such that

- There are at least k objects $\boldsymbol{o'} \in D/\{\boldsymbol{o}\}$ such that $dist(\boldsymbol{o}, \boldsymbol{o'}) \leq dist(\boldsymbol{o}, \boldsymbol{p})$.
- There are at most $k-1$ objects $\boldsymbol{o''} \in D/\{\boldsymbol{o}\}$ such that $dist(\boldsymbol{o}, \boldsymbol{o''}) < dist(\boldsymbol{o}, \boldsymbol{p})$.

In other words, $dist_k(\boldsymbol{o})$ is the distance between $\boldsymbol{o}$ and its k-nearest neighbor. Consequently, the *k-distance neighborhood* of $\boldsymbol{o}$ contains all objects of which the distance to $\boldsymbol{o}$ is not greater than $dist_k(\boldsymbol{o})$, the k-distance of $\boldsymbol{o}$, denoted by

$$N_k(\boldsymbol{o}) = \{\boldsymbol{o'} | \boldsymbol{o'} \in D, dist(\boldsymbol{o}, \boldsymbol{o'}) \leq dist_k(\boldsymbol{o})\}. \tag{11.11}$$

Note that $N_k(\boldsymbol{o})$ may contain more than k objects because multiple objects may each be the same distance away from $\boldsymbol{o}$.

We can use the average distance from the objects in $N_k(\boldsymbol{o})$ to $\boldsymbol{o}$ as the measure of the local density of $\boldsymbol{o}$. However, such a straightforward measure has a problem: If $\boldsymbol{o}$ has very close neighbors $\boldsymbol{o'}$ such that $dist(\boldsymbol{o}, \boldsymbol{o'})$ is very small, the statistical fluctuations of the distance measure can be undesirably high. To overcome this problem, we can switch to the following reachability distance measure by adding a smoothing effect.

For two objects, $\boldsymbol{o}$ and $\boldsymbol{o'}$, the *reachability distance* from $\mathbf{o}'$ to $\boldsymbol{o}$ is $dist(\mathbf{o}, \mathbf{o}')$ if $dist\ (\boldsymbol{o}, \boldsymbol{o'}) > dist_k(\boldsymbol{o})$, and $dist_k(\boldsymbol{o})$ otherwise. That is,

$$reachdist_k(\boldsymbol{o} \leftarrow \boldsymbol{o'}) = \max\{dist_k(\boldsymbol{o}), dist(\boldsymbol{o}, \boldsymbol{o'})\}. \tag{11.12}$$

Here, k is a user-specified parameter that controls the smoothing effect. Essentially, k specifies the minimum neighborhood to be examined to determine the local density of an object. Importantly, reachability distance is not symmetric, that is, in general, $reachdist_k(\boldsymbol{o} \leftarrow \boldsymbol{o}') \neq reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o})$.

Now, we can define the *local reachability density* of an object, $\boldsymbol{o}$, as

$$lrd_k(\boldsymbol{o}) = \frac{\|N_k(\boldsymbol{o})\|}{\sum_{\boldsymbol{o}' \in N_k(\boldsymbol{o})} reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o})}. \tag{11.13}$$

There is a critical difference between the density measure here for outlier detection and that in density-based clustering (Section 11.5.1). In density-based clustering, to determine whether an object can be considered a core object in a density-based cluster, we use two parameters: a radius parameter, r, to specify the range of the neighborhood, and the minimum number of points in the r-neighborhood. Both parameters are global and are applied to every object. In contrast, as motivated by the observation that relative density is the key to finding local outliers, we use the parameter k to quantify the neighborhood and do not need to specify the minimum number of objects in the neighborhood as a requirement of density. We instead calculate the local reachability density for an object and compare it with that of its neighbors to quantify the degree to which the object is considered an outlier.

Specifically, we define the *local outlier factor* of an object $\boldsymbol{o}$ as

$$LOF_k(\boldsymbol{o}) = \frac{\sum_{\boldsymbol{o}' \in N_k(\boldsymbol{o})} \frac{lrd_k(\boldsymbol{o}')}{lrd_k(\boldsymbol{o})}}{\|N_k(\boldsymbol{o})\|} = \sum_{\boldsymbol{o}' \in N_k(\boldsymbol{o})} lrd_k(\boldsymbol{o}') \cdot \sum_{\boldsymbol{o}' \in N_k(\boldsymbol{o})} reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o}). \tag{11.14}$$

In other words, the local outlier factor is the average of the ratio of the local reachability density of $\boldsymbol{o}$ and those of $\boldsymbol{o}$'s k-nearest neighbors. The lower the local reachability density of $\boldsymbol{o}$ (i.e., the smaller the item $\sum_{\boldsymbol{o}' \in N_k(\boldsymbol{o})} reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o})$) and the higher the local reachability densities of the k-nearest neighbors of $\boldsymbol{o}$, the higher the LOF value is. This exactly captures a local outlier of which the local density is relatively low compared to the local densities of its k-nearest neighbors.

The local outlier factor has some nice properties. First, for an object deep within a consistent cluster, such as the points in the center of cluster C_2 in Fig. 11.7, the local outlier factor is close to 1. This property ensures that objects inside clusters, no matter whether the cluster is dense or sparse, will not be mislabeled as outliers.

Second, for an object $\boldsymbol{o}$, the meaning of $LOF(\boldsymbol{o})$ is easy to understand. Consider the objects in Fig. 11.8, for example. For object $\boldsymbol{o}$, let

$$direct_{min}(\boldsymbol{o}) = \min\{reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o}) | \boldsymbol{o}' \in N_k(\boldsymbol{o})\} \tag{11.15}$$

be the minimum reachability distance from $\boldsymbol{o}$ to its k-nearest neighbors. Similarly, we can define

$$direct_{max}(\boldsymbol{o}) = \max\{reachdist_k(\boldsymbol{o}' \leftarrow \boldsymbol{o}) | \boldsymbol{o}' \in N_k(\boldsymbol{o})\}. \tag{11.16}$$

We also consider the neighbors of $\boldsymbol{o}$'s k-nearest neighbors. Let

$$indirect_{min}(\boldsymbol{o}) = \min\{reachdist_k(\boldsymbol{o}'' \leftarrow \boldsymbol{o}') | \boldsymbol{o}' \in N_k(\boldsymbol{o}) \text{ and } \boldsymbol{o}'' \in N_k(\boldsymbol{o}')\} \tag{11.17}$$

and

$$indirect_{max}(\boldsymbol{o}) = \max\{reachdist_k(\boldsymbol{o}'' \leftarrow \boldsymbol{o}') | \boldsymbol{o}' \in N_k(\boldsymbol{o}) \text{ and } \boldsymbol{o}'' \in N_k(\boldsymbol{o}')\}. \tag{11.18}$$

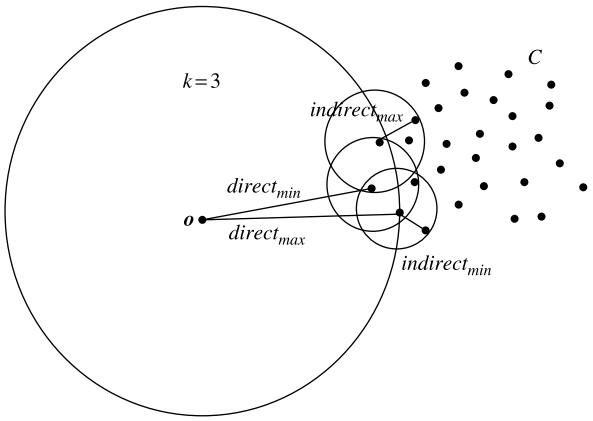


FIGURE 11.8

A property of $LOF(\boldsymbol{o})$.

Then, it can be shown that $LOF(\boldsymbol{o})$ is bounded as

$$\frac{direct_{min}(\boldsymbol{o})}{indirect_{max}(\boldsymbol{o})} \leq LOF(\boldsymbol{o}) \leq \frac{direct_{max}(\boldsymbol{o})}{indirect_{min}(\boldsymbol{o})}. \tag{11.19}$$

This result clearly shows that LOF captures the relative density of an object.

11.4 Reconstruction-based approaches

The central idea behind reconstruction-based outlier detection approaches is as follows. Since the normal data samples share certain similarities, they can often be represented in a more succinct way, compared with their original representation (e.g., an attribute vector for each data sample). With the succinct representation, we can well *reconstruct* the original representation of the normal samples. On the other hand, for samples that cannot be well reconstructed by such alternative, succinct representation, we flag them as outliers.

Example 11.14. Given a set of researchers in Fig. 11.9(a), where each researcher is described by the list of venues the corresponding author publishes. Alternatively, we use a more succinct way, namely the research areas, to represent each author, as shown in Fig. 11.9(b). For most authors, we can use such succinct representation to perfectly reconstruct their original representation in Fig. 11.9(a). For example, given the succinct representation "Data Mining" for John, Tom and Bob, we can infer (i.e., reconstruct) that their publication venues are "KDD" and "ICDM." Likewise, given the succinct representation "Software Engineering" for Van and Roy, we can infer (i.e., reconstruct) that their publication venues are "FSE" and "ICSE." Therefore we conclude that they are all "normal" researchers, which in this example means that they all focus on a specific research area (e.g., either "Data Mining" or "Software Engineering"). However, for Carl, we can only infer that his publication venues would be

Researchers	Publication Venues (Original)
John	KDD, ICDM
Tom	KDD, ICDM
Bob	KDD, ICDM
Carl	ICDM, FSE, ICSE
Van	FSE, ICSE
Roy	FSE, ICSE

(a) Original representation

Researchers	Publication Areas (Succinct)
John	Data Mining
Tom	Data Mining
Bob	Data Mining
Carl	Software Engineering
Van	Software Engineering
Roy	Software Engineering

(b) Succinct representation

FIGURE 11.9

An example of using reconstruction-based approaches to detect outlying researchers. "KDD" and "ICDM" are two conferences in the area of data mining; and "FSE" and "ICSE" are two conferences in software engineering. (a) The original representation of authors, using the publication venues. (b) The succinct representation of authors, using research areas.

"FSE" and "ICSE" based on his succinct representation ("Software Engineering"), which misses an important venue he actually publishes (i.e., "ICDM"). In other words, there is a discrepancy between Carl's original representation and reconstructed one. We say that the reconstruction quality for Carl is low. Therefore, we flag him as an outlier, which in this example means that he might be a multi-disciplinary researcher who publishes mainly "Software Engineering" but also publishes in some data mining venues (e.g., "ICDM"). Conceptually, if we treat each succinct representation (e.g., "Data Mining" vs. "Software Engineering") as a cluster, Carl is flagged as an outlier in this example, because he is the only researcher who belongs to both clusters, whereas each of the remaining researchers only belongs to a single cluster (i.e., either "Data Mining" or "Software Engineering").[2] □

The key questions in a reconstruction-based outlier detection method include (Q1) how to find the succinct representation; (Q2) how to use the succinct representation to reconstruct the original data samples; and (Q3) how to measure the quality (i.e., goodness) of reconstruction. In this section, you will learn two types of reconstruction-based outlier detection approaches, including (1) matrix-factorization based methods for numerical data and (2) pattern-based compression methods for categorical data.

11.4.1 Matrix factorization–based methods for numerical data

For data with numerical attributes (i.e., features), we represent each data sample as an attribute vector and the entire input data set is then represented by a *data matrix*, whose rows are different data samples, columns are different attributes, and entries of the data matrix indicate the attribute values of the corresponding data samples. In this setting, a powerful technique for detecting outliers is *matrix factorization*. In this method, we approximate the data matrix by two or more *low-rank* matrices, which serve as the succinct representation of the original data matrix and meanwhile, provide a natural way

[2] Notice that this is different from the clustering-based outlier detection that will be introduced in Section 11.5.1, where an outlier is often defined as a data tuple that either does not belong to any cluster or is far away from the cluster center it belongs to.

Algorithm: Matrix factorization based outlier detection

Input:

- $X_i = (X_{i,1}, ..., X_{i,m})$ $(i = 1, ..., n)$, n data samples each of which is represented by an m dimensional numerical attribute vector;
- r, the rank;
- k, the number of outliers.

Output: A set of k outliers.
Method:

//each row of $\boldsymbol{X}$ is a data sample
(1) Represent the input data samples as an $n \times m$ data matrix $\boldsymbol{X}$;
(2) Approximate the data matrix by two rank r-matrices: $\boldsymbol{X} \approx \boldsymbol{F}\boldsymbol{G}$;
(3) **for** $(i = 1, ..., n)\{$ // for each sample
//compute the reconstruction error
//$\boldsymbol{F}(i, j)$ is the element of $\boldsymbol{F}$ at the i^{th} row and the j^{th} column
//$\boldsymbol{G}(j, :)$ is the j^{th} row of $\boldsymbol{G}$
(4) Compute $r_i = \|X_i - \hat{X}_i\|^2 = \|X_i - \sum_{j=1}^{r} \boldsymbol{F}(i, j)\boldsymbol{G}(j, :)\|^2$;
$\}$
(5) Return k samples with the largest reconstruction errors r_i $(i = 1, ..., n)$.

FIGURE 11.10

Matrix factorization–based outlier detection.

to reconstruct the original data matrix. The *reconstruction error* of each data sample is used as an indicator of the outlier-ness: the higher the reconstruction error, the more likely the given data sample is an outlier. The algorithm is summarized in Fig. 11.10 and the details are depicted as follows.

We start with representing the input data samples in the form of a data matrix $\boldsymbol{X}$ (step 1), whose rows represent data samples and columns are attributes (i.e., features). Then (step 2), we approximate the data matrix $\boldsymbol{X}$ by the multiplication of two matrices $\boldsymbol{F}$ and $\boldsymbol{G}$. Here, an important point is that the rank of $\boldsymbol{F}$ and $\boldsymbol{G}$ should be much smaller than the data matrix (i.e., $r \ll m$). In this way, the r rows of matrix $\boldsymbol{G}$ provide a more succinct way to represent the input data samples, that is, $\boldsymbol{G}(j, :)$ $(j = 1, ..., r)$ are the new, succinct representation. In the meanwhile, matrix $\boldsymbol{F}$ tells us how to use such succinct representation to *reconstruct* the original data sample, that is, $\hat{X}_i = \sum_{j=1}^{r} \boldsymbol{F}(i, j)\boldsymbol{G}(j, :)$, where $\boldsymbol{F}(i, j)$ is the element of $\boldsymbol{F}$ at the ith row and the jth column. The squared distance between the original data sample X_i and the reconstructed one $\hat{X}_i$ is called *reconstruction error* (step 4), and it measures the outlier-ness of the corresponding data sample. Finally, in step 5, we return k data samples with the highest reconstruction errors as the outliers.

Example 11.15. For Example 11.14, we represent each author as a 4-D binary vector, indicating if the author publishes in the corresponding conferences, where the four dimensions are the four conferences, including "KDD," "ICDM," "FSE," and "ICSE," respectively. For example, John is represented as $(1, 1, 0, 0)$, meaning that he publishes in both "KDD" and "ICDM"; Carl is represented as $(0, 1, 1, 1)$, meaning that he publishes in all conferences but "KDD." We organize the feature vectors of all six authors as a 6×4 data matrix $\boldsymbol{X}$ (the left part of Fig. 11.11). Then, we approximate the data matrix $\boldsymbol{X}$ by the multiplication of two low-rank matrices $\boldsymbol{F}$ and $\boldsymbol{G}$, both with a rank of 2 (the middle part of Fig. 11.11). The two rows of matrix $\boldsymbol{G}$ provide an alternative, succinct representation compared with the original four features in the data matrix $\boldsymbol{X}$. For example, the first row of

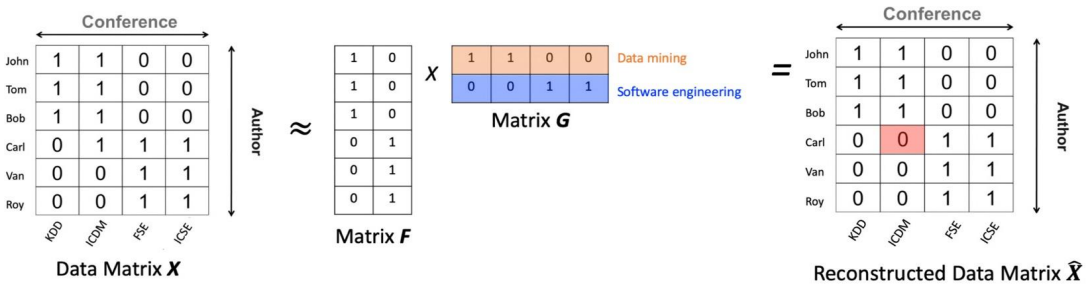


FIGURE 11.11

Matrix factorization based method to detect outliers for Example 11.14. By approximating the original data matrix (on the left) by the multiplication of two low-rank matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ (in the middle), it provides an alternative, more succinct representation of the original features. The reconstruction error of each author by comparing the original and the reconstructed representation (on the right) provides an indicator of the outlier-ness of each author. In this example, the reconstruction errors for all researchers except Carl is 0, and thus Carl is flagged as an outlying author.

$\boldsymbol{G}$ represents data mining research area, and the second row of $\boldsymbol{G}$ represents the software engineering research area. With the help of the rows of $\boldsymbol{F}$, we can use the succinct representation in $\boldsymbol{G}$ to reconstruct the original representation of each author. For example, the reconstructed representation for John is $1 \times (1, 1, 0, 0) + 0 \times (0, 0, 1, 1) = (1, 1, 0, 0)$, and the reconstructed representation for Carl is $0 \times (1, 1, 0, 0) + 1 \times (0, 0, 1, 1) = (0, 0, 1, 1)$. We put the reconstructed representation of all six authors in another matrix $\hat{\boldsymbol{X}}$ (the right part of Fig. 11.11). Then, we compute the squared distance between the rows of the original data matrix $\boldsymbol{X}$ and that of the reconstructed data matrix $\hat{\boldsymbol{X}}$, which are used as the outlier-ness scores of the corresponding authors. In this example, the reconstruction error is zero for all authors but Carl, who has a reconstruction error of 1. Therefore we flag Carl as an outlying author. □

"So, how can we find these two magic low-rank matrices $\boldsymbol{F}$ and $\boldsymbol{G}$?" Many approaches exist. A popular choice is to use *singular value decomposition* (SVD). Given an input data matrix $\boldsymbol{X}$, SVD approximates it by the multiplication of three rank-r matrices

$$\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}', \tag{11.20}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are *orthonormal matrices* and their columns are called left singular vectors and right singular vectors of data matrix $\boldsymbol{X}$, respectively, $\boldsymbol{V}'$ is the transpose of matrix $\boldsymbol{V}$, and $\boldsymbol{\Sigma}$ is a diagonal with nonnegative diagonal elements called singular values. For the purpose of outlier detection, we can set $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}$ and $\boldsymbol{G} = \boldsymbol{V}'$. See Fig. 11.12 for an illustration.

The full mathematical details of SVD are outside the scope of this textbook. Many software packages exist to compute SVD for a given data matrix. An appealing property of SVD lies in its *optimal approximation*. This means that among all rank-r matrices, SVD provides the best approximation of the original data matrix in terms of both L_2 norm and Frobenious norm. Another interesting property of SVD lies in its close connection with PCA. In particular, if the data matrix $\boldsymbol{X}$ is "centered," meaning that each column (i.e., feature) of $\boldsymbol{X}$ has a zero sample mean (we can achieve this, say by z-score normalization), the columns of matrix $\boldsymbol{V}$ (i.e., the right singular vectors) are exactly the first r principle components of the input data. (Why this is true is left as an exercise.)

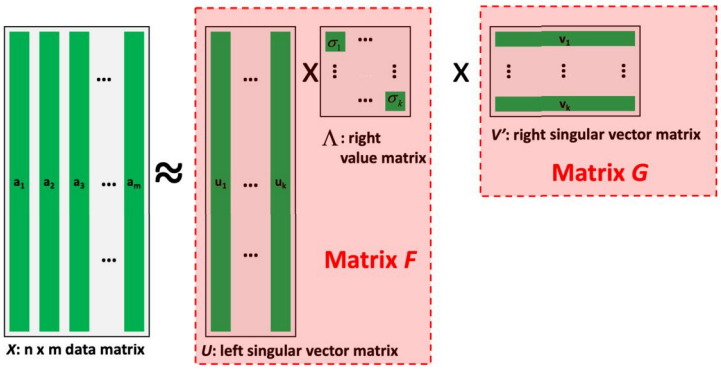


FIGURE 11.12

An illustration of SVD. Given a data matrix $\boldsymbol{X}$ (on the left), SVD approximates it by the multiplication of three rank-r matrices. The multiplication of the left singular vector matrix $\boldsymbol{U}$ and the singular value matrix $\boldsymbol{\Sigma}$ becomes matrix $\boldsymbol{F}$ in Fig. 11.11 and the transpose of the right singular vector matrix $\boldsymbol{V}'$ becomes matrix $\boldsymbol{G}$ in Fig. 11.11. When the data matrix $\boldsymbol{X}$ is centered (i.e., each of its columns has zero sample mean), the columns of $\boldsymbol{V}$ are the first r principal components of $\boldsymbol{X}$.

A potential limitation of SVD is that the low-rank matrices ($\boldsymbol{F}$ and $\boldsymbol{G}$) are often dense, even when the input data matrix $\boldsymbol{X}$ itself is sparse, which makes the detection results somehow hard for the end user to interpret. To address this issue, *example-based matrix factorization* has been developed. For instance, CX-decomposition first samples (with replacement) a few columns of the data matrix $\boldsymbol{X}$ to form matrix $\boldsymbol{F}$, and then finds matrix $\boldsymbol{G}$ by projecting the data matrix $\boldsymbol{X}$ onto the column space spanned by matrix $\boldsymbol{F}$.[3] In CX-decomposition, there might exist redundancy among the columns in matrix $\boldsymbol{F}$ with duplicated or linearly correlated columns that have no impact on improving the reconstruction error (the indicator for outlier detection) yet waste both time and space. An improved example-based matrix factorization called *Colibri* only uses linearly independent columns of the data matrix $\boldsymbol{X}$ to construct the matrix $\boldsymbol{F}$. Both CX-decomposition and Colibri were found to improve the efficiency and interpretability of outlier detection.

Example 11.16. Suppose we are given a set of authors, each of whom is represented by how many papers she has published in two conferences, including "KDD"—a data mining conference, and "ISBM"—a computational biology conference. Therefore the input data matrix $\boldsymbol{X}$ is an $n \times 2$ matrix, where each row (i.e., a circle in Fig. 11.13(a)) is an author and n is the number of authors. If we use SVD to factorize the data matrix $\boldsymbol{X} \approx \boldsymbol{F}\boldsymbol{G}$, each column of matrix $\boldsymbol{G}$ is a right singular vector of $\boldsymbol{X}$ that is a linear combination of *all* rows of the data matrix (Fig. 11.13(b)). Therefore the resulting matrix $\boldsymbol{G}$ is dense, and so is matrix $\boldsymbol{F}$. In contrast, example-based factorization (e.g., CX-decomposition and Colibri in Fig. 11.13(c-d)) uses actual data samples (e.g., sampled authors) to construct matrix $\boldsymbol{G}$, which is more interpretable and computationally more efficient than SVD. In order to detect outlying authors,

[3] Mathematically, let $\boldsymbol{F}$ consist of columns sampled from the data matrix $\boldsymbol{X}$. We approximate $\boldsymbol{X}$ as $\boldsymbol{X} \approx \boldsymbol{F}(\boldsymbol{F}'\boldsymbol{F})^{+}\boldsymbol{F}'\boldsymbol{X}$, where $^{+}$ is the matrix pseudo-inverse. We set matrix $\boldsymbol{G} = (\boldsymbol{F}'\boldsymbol{F})^{+}\boldsymbol{F}'\boldsymbol{X}$.

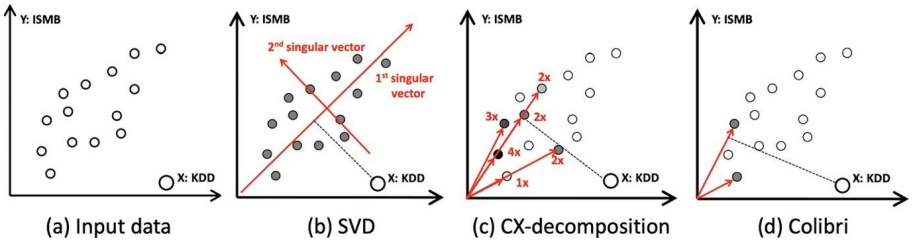


FIGURE 11.13

A pictorial comparison between SVD, CX-decomposition, and Colibri. (a) Input data: each circle is an author who is represented by two features ("KDD" and "ISMB"). (b) SVD finds optimal factorization, where the two right singular vectors (two red (gray in print version) arrows), together with the two corresponding singular values, form the matrix $\boldsymbol{G}$. Each right singular vector is a linear combination of all input authors, resulting in a dense matrix $\boldsymbol{G}$. (c) CX-decomposition randomly samples some actual authors (gray circles) to form matrix $\boldsymbol{G}$, and some sampled columns might be duplicated (indicated by the number of the corresponding circle). (d) Colibri uses two linearly independent, sampled authors (two gray circles) to form the matrix $\boldsymbol{G}$. Using the first singular vector (b), or one sampled data point by either CX-decomposition (c) or Colibri (d), we might flag the author on the right bottom corner as an outlier, with a high reconstruction error (the dashed line).

we can use the first singular vector (b), or one sampled data point by either CX-decomposition (c) or Colibri (d) as the succinct representation. Then, an author (e.g., the one on the right bottom corner) with a high reconstruction error (indicated by the length of the dashed line in the figure) is flagged as an outlier. Notice that the meaning of outliers found in this example is quite different from Example 11.14. In Example 11.14, most (i.e., normal) authors belong to one of the two research areas ("Data Mining" or "Software Engineering"), and thus Carl, who belongs to both areas, is flagged as an outlying author. In contrast, in this example, for most (i.e., normal) authors, the numbers of publications in "ISMB" are positively correlated with that in "KDD," which is captured by the succinct representation (e.g., the first singular vector). Therefore the author on the right bottom corner, with a significant number of publications in "KDD" yet almost none in "ISMB," is flagged as an outlier. □

When the input data matrix $\boldsymbol{X}$ is *nonnegative*, a natural choice is to use nonnegative matrix factorization (NMF, which was introduced in Chapter 9). In general, nonnegative entries are easier to interpret since they often correspond to the actual "parts" of the input data (e.g., a nose, an eye if the input data is face image). A typical NMF method imposes the nonnegativity constraints on the two factorized matrices ($\boldsymbol{F}$ and $\boldsymbol{G}$). However, in Fig. 11.11, we use the *residual matrix* $\boldsymbol{R} = \boldsymbol{X} - \boldsymbol{F}\boldsymbol{G}$ to detect outlying data samples (e.g., if the squared norm of a row of the residual matrix $\boldsymbol{R}$ is high, the corresponding data are flagged as an outlier). Therefore it is natural to require the residual matrix $\boldsymbol{R}$, instead of the factorized matrices $\boldsymbol{F}$ and $\boldsymbol{G}$, to be nonnegative. Matrix factorization with this type of constraint (i.e., the entries of the residual matrix must be nonnegative) is called *nonnegative residual matrix factorization* (NrMF). NrMF was found to be more interpretable to detect outliers which corresponds to some actual behaviors or activities of certain data samples.

Example 11.17. An IP source might be detected as a suspicious port-scanner if it *sends* packages to a lot of destinations in an IP traffic network; we might flag a group of users who always give good ratings

to another group of users as a collusion type of fake reviewers. If we map such behaviors or activities (e.g., "sends packages," "gives good ratings," etc.) to the language of matrix factorization, it suggests that the corresponding entries in the residual matrix should be nonnegative. □

Another method, called *robust PCA*, requires that the residual matrix **R** to be sparse with most of the rows being empty. The intuition is as follows. For the vast majority of normal data tuples, they can be perfectly reconstructed by the factorized matrices, whereas for a small number of outlying tuples, they bear nonzero reconstruction errors indicated by the corresponding, nonempty rows in the residual matrix **R**.

Matrix factorization based methods are linear methods in that the new, more succinct representation is a linear combination of the original features. For example, in SVD, each right singular vector is a linear combination of the original features of the input data. For the example in Fig. 11.9, the new representation "Data Mining" is the result of a linear combination of the two conferences including "KDD" and "ICDM." If we wish to capture the *nonlinear* relationship between the original features, we can use *autoencoder*, an unsupervised deep learning architecture introduced in Chapter 10, to reconstruct the original data samples. Again, the data samples with high reconstruction errors are flagged as outliers.

11.4.2 Pattern-based compression methods for categorical data

For input data with categorical attributes, we could convert the categorical attributes to binary attributes and then apply the matrix factorization based methods to detect outliers.

Example 11.18. Given an input data set in Fig. 11.14, where each row represents a customer who is described by three categorical attributes, namely "income," "credit," and "purchase." In order to detect abnormal customers, we first convert the input data set into a binary data matrix $\boldsymbol{B}$ in Fig. 11.15(a). Since each categorical attribute has three possible values, including "High," "Medium," and "Low," it is converted into three binary attributes. (How to convert a categorical attribute to binary ones was introduced in Chapter 2.) Next, we approximate the binary data matrix $\boldsymbol{B}$ by the multiplication of two low-rank matrices, say the ones in Fig. 11.15(b), which are in turn used to reconstruct the original data matrix $\boldsymbol{B}$. By comparing the original and reconstructed data matrices, we find out that the reconstruction errors for the first four customers are all zeros. On the other hand, the reconstruction errors for Tom and Jim are 2.5 and 1.34, respectively, both of whom are thus flagged as outliers. □

	Income	Credit	Purchase
John	High	High	High
Amy	High	High	High
Carl	Low	Low	Low
Mary	Low	Low	Low
Tom	High	Low	Medium
Jim	High	Low	Low

FIGURE 11.14

Input data with categorical attributes, including "Income," "Credit," and "Purchase." Each categorical attribute has three values, including "High," "Medium," and "Low."

	I/H	I/M	I/L	C/H	C/M	C/L	P/H	P/M	P/L
John	1	0	0	1	0	0	1	0	0
Amy	1	0	0	1	0	0	1	0	0
Carl	0	0	1	0	0	1	0	0	1
Mary	0	0	1	0	0	1	0	0	1
Tom	1	0	0	0	0	1	0	1	0
Jim	1	0	0	0	0	1	0	0	1

(a) Representing the input categorical data as binary matrix $\boldsymbol{B}$

1	0
1	0
0	1
0	1
0.5	0.5
0.3	0.7

Matrix $\boldsymbol{F}$

X

1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1

Matrix $\boldsymbol{G}$

(b) The two factorized matrices

	I/H	I/M	I/L	C/H	C/M	C/L	P/H	P/M	P/L
John	1	0	0	1	0	0	1	0	0
Amy	1	0	0	1	0	0	1	0	0
Carl	0	0	1	0	0	1	0	0	1
Mary	0	0	1	0	0	1	0	0	1
Tom	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5
Jim	0.3	0	0.7	0.3	0	0.7	0.3	0	0.7

(c) The reconstructed matrix

FIGURE 11.15

Using matrix factorization based method to detect outliers of input data with categorical attributes. (a) Representing the input data by a binary data matrix $\boldsymbol{B}$ by converting the categorical attributes to binary attributes. (b) The two factorized matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ to approximate the binary data matrix $\boldsymbol{B}$. "I/H," "I/M," and "I/L" represent the attribute values of "Income" being "High," "Medium," and "Low," respectively. Likewise. "C/H," "C/M," and "C/L" represent the attribute values of "Credit" being "High," "Medium," and "Low," respectively; and "P/H," "P/M," and "P/L" represent the attribute values of "Purchase" being "High," "Medium," and "Low," respectively. (c) The reconstructed matrix.

When a categorical attribute has many different values, the binary data matrix $\boldsymbol{B}$ would have many columns since we need to create a separate column (i.e., binary feature) for each categorical attribute value. This might make the matrix factorization process computationally intensive and the detection results hard to interpret. For example, it might be hard to tell why "Tom" and "Jim" are flagged as outliers by just looking at the factorized matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ in Fig. 11.15(b). *So, how can we do better?*

An alternative method is to use **pattern-based compression** methods to detect outliers for input data with categorical attributes.

For the input data set in Fig. 11.14, we can construct a *code table* (also known as *dictionary*), shown in Fig. 11.16. At the first glance, the table might look a bit abstract to you. Let us explain the details.

The first column of Fig. 11.16 consists of a set of *code words*, also known as a *pattern*. Each code word or pattern (e.g., "[I/H, C/H, P/H]") consists of one or more *items*, which are essentially an attribute-value pairs (e.g., "I/H" meaning the value of "Income" is "High"). (Recall that we have used the similar notations for pattern-based classification in Chapter 7.) The second column of Fig. 11.16 contains the binary codes that are used to represent the corresponding code words, and the last column is

Code word	Code	Usage	Code Length
[I/H, C/H, P/H]	01	2	2
[I/L, C/L, P/L]	10	2	2
[I/H, C/L]	11	2	2
[P/M]	001	1	3
[P/L]	010	1	3

FIGURE 11.16

Code table for the input data in Fig. 11.14.

simply the length of each code (i.e., how many bits are used in the corresponding code).[4] Using the code words in the first column, we can now represent the original input data samples in an alternative, more succinct way. For example, for John and Amy, we can just use the first code word "[I/H, C/H, P/H]" to represent them respectively. Likewise, for Carl and Mary, we use the second code word "[I/L, C/L, P/L]" to represent them respectively. On the other hand, for Tom, we need two code words, including "[I/H, C/L]" and "[P/M]," to represent him; and for Jim, we also need two code words to represent him, including "[I/H, C/L]" and "[P/L]." If a given code word is *used* to represent a data sample, we say that the data sample *covers* the corresponding code word. The total number of times that a given code word is used to represent the entire data set is the *usage* of that code word (i.e., the third column of Fig. 11.16). For example, the usage of "[I/H, C/H, P/H]" is 2, since it is used to represent both John and Amy; the usage of "[P/M]" is one since it is only used to represent Tom.

Once we have the alternative representation of the input data sample, we can use the *encoding length* as an indicator of the outlier-ness. The longer the encoding length, the more likely the given data sample is an outlier. For the example in Fig. 11.14, John is represented by a single code word "[I/H, C/H, P/H]" with the encoding length of 2 (i.e., "01"). Likewise, the encoding length for Amy, Carl and Mary is also 2. On the other hand, Tom is represented by two code words, including "[I/H, C/L]" and "[P/M]" with the total encoding length of 5; Jim is also represented by two code words, including "[I/H, C/L]" and "[P/L]" with the total encoding length of 5. Therefore we flag both Tom and Jim as outlying customers. Compared with matrix-factorization methods, the pattern-compression based methods have an advantage for detecting outliers with categorical attributes in terms of interpretability. For example, by looking at the code words that Tom and Jim cover, an atypical pattern, namely "[I/H, C/L]" (high income but low credit score), stands out. Therefore the low purchase activity of Tom and Jim might be attributed to their low credit score despite that the income of both of them is high.

But, you might wonder, how can we obtain the code table from the input data set? Why is certain code word (e.g., "[I/H, C/H, P/H]") assigned by a shorter code than others (e.g., "[I/H, C/L]," "[P/L]")? Why is the encoding length a reasonable indicator of outlier-ness? Pattern-based compression methods often use *Minimum Description Length* (MDL) principle to search for the optimal code table. The full mathematical details of MDL are outside the scope of the textbook and interested readers can refer to the bibliographic notes. Simply put, MDL principles favors the model that can most

[4] It turns out that for the outlier detection task, it is the length of codes, not the actual codes, of the code words that matters with the detection results.

succinctly describe the input data set. That is, it favors a model M that minimizes the following cost:

$$L(M) + L(D|M), \tag{11.21}$$

where $L(M)$ is the length in bits of the description of the model itself, and $L(D|M)$ is the length in bits of the description of the data set D using the given model M.

In the outlier detection setting, the models are all possible code tables and thus $L(M)$ is the encoding cost to describe the first two columns in Fig. 11.16, including the encoding cost of the actual code words (e.g., "[I/H, C/H, P/H]") and the corresponding codes (e.g., "01"). Intuitively, in order to minimize $L(M)$, we would favor a concise code table (e.g., with a small number of short code words). $L(D|M)$ is the total cost to describe the input data set using the given code table M, which is the summation of the encoding length of each data sample using the given code table. Intuitively, in order to minimize $L(D|M)$, on the one side, we should choose a comprehensive set of code words so that each input data sample can be represented by a subset of chosen code words. On the other hand, we should choose frequent code words, and if a code word is frequently *used* to represent different input data samples, we should assign it with a shorter code.[5] In this way, the total encoding cost $L(D|M)$ could be minimized. Therefore if the encoding length of a given data sample is long, it suggests that it is represented by rare, infrequent patterns (code words), and thus looks outlying.

11.5 Clustering- vs. classification-based approaches

Depending on the availability of *supervision*, which is in the form of the labels of training tuples regarding whether they are normal or outlying samples, we can categorize outlier detection techniques into **clustering-based** vs. **classification-based** methods. Clustering-based approaches are unsupervised methods, which detect outliers by examining the relationship between objects and clusters. Intuitively, an outlier could be an object that belongs to a small and remote cluster or does not belong to any cluster. The notion of outliers is highly related to that of clusters. Classification-based approaches are essentially supervised methods, which treat outlier detection as a classification problem. The general idea of classification-based outlier detection methods is to train a classification model that can distinguish normal data from outliers.

11.5.1 Clustering-based approaches

Generally speaking, there are three approaches to clustering-based outlier detection. Consider an object.

- Does the object belong to any cluster? If not, then it is identified as an outlier.
- Is there a large distance between the object and the cluster to which it is closest? If yes, it is an outlier.
- Is the object part of a small or sparse cluster? If yes, then all the objects in that cluster are outliers.

[5] According to Shannon entropy, the optimal encoding length of a given code c is $\log \frac{\sum_i \text{usage}(i)}{\text{usage}(c)}$. The higher the usage of the code c, the shorter its encoding length.

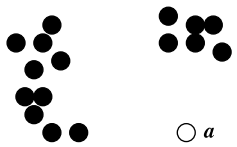


FIGURE 11.17

Object ***a*** is an outlier because it does not belong to any cluster.

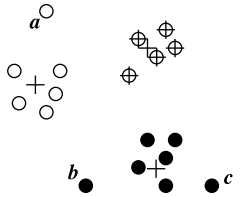


FIGURE 11.18

Outliers (***a***, ***b***, ***c***) are far from the clusters to which they are closest (with respect to the cluster centers).

Let's look at examples of each of these approaches.

Example 11.19. Detecting outliers as objects that do not belong to any cluster. Gregarious animals (e.g., goats and deer) live and move in flocks. Using outlier detection, we can identify outliers as animals that are not part of a flock. Such animals may be either lost or wounded.

In Fig. 11.17, each point represents an animal living in a group. Using a density-based clustering method, such as DBSCAN, we note that the black points belong to clusters. The white point, ***a***, does not belong to any cluster, and thus is declared an outlier. □

The second approach to clustering-based outlier detection considers the distance between an object and the cluster to which it is closest. If the distance is large, then the object is likely an outlier with respect to the cluster. Thus this approach detects individual outliers with respect to clusters.

Example 11.20. Clustering-based outlier detection using distance to the closest cluster. Using the k-means clustering method, we can partition the data points shown in Fig. 11.18 into three clusters, as shown using different symbols. The center of each cluster is marked with a $+$.

For each object ***o***, we can assign an outlier-ness score according to the distance between the object and the center that is closest to the object. Suppose the closest center to ***o*** is $\boldsymbol{c}_o$; then the distance between ***o*** and $\boldsymbol{c}_o$ is $dist(\boldsymbol{o}, \boldsymbol{c}_o)$, and the average distance between $\boldsymbol{c}_o$ and the objects assigned to $\boldsymbol{c}_o$ is $l_{\boldsymbol{c}_o}$. The ratio $\frac{dist(\boldsymbol{o},\boldsymbol{c}_o)}{l_{\boldsymbol{c}_o}}$ measures how $dist(\boldsymbol{o}, \boldsymbol{c}_o)$ stands out from the average. The larger the ratio, the farther away ***o*** is relative from the center, and the more likely ***o*** is an outlier. In Fig. 11.18, points ***a***, ***b***, and ***c*** are relatively far away from their corresponding centers and thus are suspected of being outliers. □

This approach can also be used for intrusion detection, as described in Example 11.21.

Example 11.21. Intrusion detection by clustering-based outlier detection. A bootstrap method was developed to detect intrusions in TCP connection data by considering the similarity between data points and the clusters in a training data set. The method consists of three steps.

1. A training data set is used to find patterns of normal data. Specifically, the TCP connection data are segmented according to, say, dates. Frequent itemsets are found in each segment. The frequent itemsets that are in a majority of the segments are considered as patterns of normal data and are referred to as "base connections."
2. Connections in the training data that contain base connections are treated as attack-free. Such connections are clustered into groups.
3. The data points in the original data set are compared with the clusters mined in step 2. Any point that is deemed an outlier with respect to the clusters is declared as a possible attack. □

Note that each of the approaches we have seen so far detects only individual objects as outliers because they compare objects one at a time against clusters in the data set. However, in a large data set, some outliers may be similar and form a small cluster. In intrusion detection, for example, hackers who use similar tactics to attack a system may form a cluster. The approaches discussed so far may be deceived by such outliers.

To overcome this problem, a third approach to cluster-based outlier detection identifies small or sparse clusters and declares the objects in those clusters to be outliers as well. An example of this approach is the *FindCBLOF* algorithm, which works as follows.

1. Find clusters in a data set and sort them according to decreasing size. The algorithm assumes that most of the data points are not outliers. It uses a parameter α ($0 \leq \alpha \leq 1$) to distinguish large from small clusters. Any cluster that contains at least a percentage α (e.g., $\alpha = 30\%$) of the data set is considered a "large cluster." The remaining clusters are referred to as "small clusters."
2. To each data point, assign a *cluster-based local outlier factor* (CBLOF). For a point belonging to a large cluster, its CBLOF is the product of the cluster's size and the similarity between the point and the cluster. For a point belonging to a small cluster, its CBLOF is calculated as the product of the size of the small cluster and the similarity between the point and the closest large cluster.

CBLOF defines the similarity between a point and a cluster in a statistical way that represents the probability that the point belongs to the cluster. The larger the value, the more similar the point and the cluster are. The CBLOF score can detect outlier points that are far from any clusters. In addition, small clusters that are far from any large cluster are considered to consist of outliers. The points with the lowest CBLOF scores are suspected outliers.

Example 11.22. Detecting outliers in small clusters. The data points in Fig. 11.19 form three clusters: large clusters, C_1 and C_2, and a small cluster, C_3. Object $\boldsymbol{o}$ does not belong to any cluster.

Using CBLOF, *FindCBLOF* can identify $\boldsymbol{o}$ and the points in cluster C_3 as outliers. For $\boldsymbol{o}$, the closest large cluster is C_1. The CBLOF score is simply the similarity between $\boldsymbol{o}$ and C_1, which is small. For the points in C_3, the closest large cluster is C_2. Although there are three points in cluster C_3, the similarity between those points and cluster C_2 is low, and $|C_3| = 3$ is small; thus, the CBLOF scores of points in C_3 are small. □

Clustering-based approaches may incur high computational costs if they have to find clusters before detecting outliers. Several techniques have been developed for improved efficiency. For example, **fixed-width clustering** is a linear-time technique that is used in some outlier detection methods. The idea is

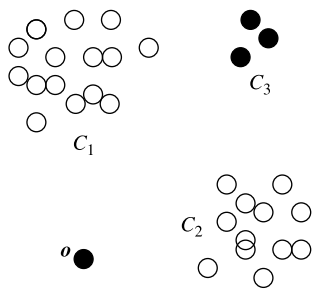


FIGURE 11.19

Outliers in small clusters.

simple yet efficient. A point is assigned to a cluster if the center of the cluster is within a predefined distance threshold from the point. If a point cannot be assigned to any existing cluster, a new cluster is created. The distance threshold may be learned from the training data under certain conditions.

Clustering-based outlier detection methods have the following advantages. First, they can detect outliers without requiring any labeled data, that is, in an unsupervised way. They work for many data types. Clusters can be regarded as summaries of the data. Once the clusters are obtained, clustering-based methods need only compare any object against the clusters to determine whether the object is an outlier. This process is typically fast because the number of clusters is usually small compared to the total number of objects.

A weakness of clustering-based outlier detection is its effectiveness, which highly depends on the clustering method used. Such methods may not be optimized for outlier detection. Clustering methods are often costly for large data sets, which could become a bottleneck.

11.5.2 Classification-based approaches

Let us consider a training set that contains samples labeled as "normal" and others labeled as "outlier." A classifier can then be constructed based on the training set. Any classification method can be used (Chapters 6 and 7). This kind of brute-force approach, however, does not work well for outlier detection because the training set is typically heavily biased. That is, the number of normal samples likely far exceeds the number of outlier samples. This imbalance, where the number of outlier samples may be insufficient, can prevent us from building an accurate classifier. Consider intrusion detection in a system, for example. Because most system accesses are normal, it is easy to obtain a good representation of the normal events. However, it is infeasible to enumerate all potential intrusions, as new and unexpected attempts occur from time to time. Hence, we are left with an insufficient representation of the outlier (or intrusion) samples.

To overcome this challenge, classification-based outlier detection methods often use a *one-class model*. That is, a classifier is built to describe only the normal class. Any samples that do not belong to the normal class are regarded as outliers. For example, in **one-class SVM**, it seeks to find a max-margin hyperplane in the transformed high-dimensional space (called reproducing kernel Hilbert space or RKHS for short) that separates the normal data tuples and the origin that represents the outlier. In

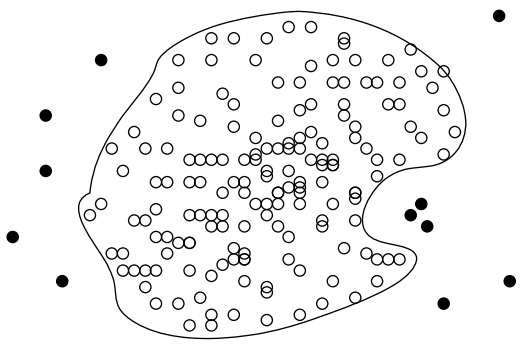


FIGURE 11.20

Learning a model for the normal class.

Support Vector Data Description (SVDD), it seeks to find a minimum hypersphere that contains all the normal tuples.

Example 11.23. Outlier detection using a one-class model. Consider the training set shown in Fig. 11.20, where white points are samples labeled as "normal" and black points are samples labeled as "outlier." To build a model for outlier detection, we can learn the decision boundary of the normal class using classification methods such as SVM (Chapter 7), as illustrated. Given a new object, if the object is within the decision boundary of the normal class, it is treated as a normal case. If the object is outside the decision boundary, it is declared an outlier.

An advantage of using only the model of the normal class to detect outliers is that the model can detect new outliers that may not appear close to any outlier objects in the training set. This occurs as long as such new outliers fall outside the decision boundary of the normal class. □

The idea of using the decision boundary of the normal class can be extended to handle situations where the normal objects may belong to multiple classes. For example, an electronics store accepts returned items. Customers can return items for a number of reasons (corresponding to class categories) such as "product design defects" and "product damaged during shipment." Each such class is regarded as normal. To detect outlier cases, the store can learn a model for each normal class. To determine whether a case is an outlier, we can run each model on the case. If the case does not fit any of the models, then it is declared an outlier.

Classification-based methods and clustering-based methods can be further combined to detect outliers in a semisupervised learning way.

Example 11.24. Outlier detection by semisupervised learning. Consider Fig. 11.21, where objects are labeled as either "normal" or "outlier" or have no label at all. Using a clustering-based approach, we find a large cluster, C, and a small cluster, C_1. Because some objects in C carry the label "normal," we can treat all objects in this cluster (including those without labels) as normal objects. We use the one-class model of this cluster to identify normal objects in outlier detection. Similarly, because some objects in cluster C_1 carry the label "outlier," we declare all objects in C_1 as outliers. Any object that does not fall into the model for C (e.g., $\boldsymbol{a}$) is considered an outlier as well. □

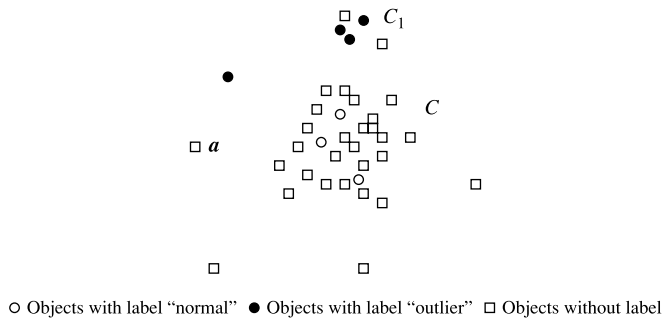


FIGURE 11.21

Detecting outliers by semisupervised learning.

Classification-based methods can incorporate human domain knowledge into the detection process by learning from the labeled samples. Once the classification model is constructed, the outlier detection process is fast. It only needs to compare the objects to be examined against the model learned from the training data. The quality of classification-based methods heavily depends on the availability and quality of the training set. In many applications, it is difficult to obtain representative and high-quality training data, which limits the applicability of classification-based methods. A promising solution is to combine outlier detection and active learning (introduced in Chapter 7) to obtain a few labeled training tuples for each class (e.g., normal class, outlying class), with the help of a human annotator.

11.6 Mining contextual and collective outliers

An object in a given data set is a **contextual outlier** (or *conditional outlier*) if it deviates significantly with respect to a specific context of the object (Section 11.1). The context is defined using **contextual attributes**. These depend heavily on the application and are often provided by users as part of the contextual outlier detection task. Contextual attributes can include spatial attributes, time, network locations, and sophisticated structured attributes. In addition, **behavioral attributes** define characteristics of the object and are used to evaluate whether the object is an outlier in the context to which it belongs.

Example 11.25. Contextual outliers. To determine whether the temperature of a location is exceptional (i.e., an outlier), the attributes specifying information about the location can serve as contextual attributes. These attributes may be spatial attributes (e.g., longitude and latitude) or location attributes in a graph or network. The attribute *time* can also be used. In customer-relationship management, whether a customer is an outlier may depend on other customers with similar profiles. Here, the attributes defining customer profiles provide the context for outlier detection. □

In comparison to outlier detection in general, identifying contextual outliers requires analyzing the corresponding contextual information. Contextual outlier detection methods can be divided into two categories according to whether the contexts can be clearly identified.

11.6.1 Transforming contextual outlier detection to conventional outlier detection

This category of methods is for situations where the contexts can be clearly identified. The idea is to transform the contextual outlier detection problem into a typical outlier detection problem. Specifically, for a given data object, we can evaluate whether the object is an outlier in two steps. In the first step, we identify the context of the object using the contextual attributes. In the second step, we calculate the outlier-ness score for the object in the context using a conventional outlier detection method.

Example 11.26. Contextual outlier detection when the context can be clearly identified. In customer-relationship management, we can detect outlier customers in the context of customer groups. Suppose an electronics store maintains customer information on four attributes, namely *age_group* (i.e., under 25, 25–45, 45–65, and over 65), *postal_code*, *number_of_transactions_per_year*, and *annual_total_transaction_amount*. The attributes *age_group* and *postal_code* serve as contextual attributes, and the attributes *number_of_transactions_per_year* and *annual_total_transaction_amount* are behavioral attributes. □

To detect contextual outliers in this setting, for a customer, $\boldsymbol{c}$, we can first locate the context of $\boldsymbol{c}$ using the attributes *age_group* and *postal_code*. We can then compare $\boldsymbol{c}$ with the other customers in the same group and use a conventional outlier detection method, such as some of the ones discussed earlier, to determine whether $\boldsymbol{c}$ is an outlier.

Contexts may be specified at different levels of granularity. Suppose the same electronics store also maintains customer information at a more detailed level for the attributes *age*, *postal_code*, *number_of_transactions_per_year*, and *annual_total_transaction_amount*. We can still group customers on *age* and *postal_code* and then mine outliers in each group. What if the number of customers falling into a group is very small or even zero? For a customer, $\boldsymbol{c}$, if the corresponding context contains very few or even no other customers, the evaluation of whether $\boldsymbol{c}$ is an outlier using the exact context is unreliable or even impossible.

To overcome this challenge, we can assume that customers of similar age and who live within the same area should have similar normal behavior. This assumption can help to generalize contexts and makes for more effective outlier detection. For example, using a set of training data, we may learn a mixture model, U, of the data on the contextual attributes, and another mixture model, V, of the data on the behavior attributes. A mapping $p(V_i|U_j)$ is also learned to capture the probability that a data object $\boldsymbol{o}$ belonging to cluster U_j on the contextual attributes is generated by cluster V_i on the behavior attributes. The outlier-ness score can then be calculated as

$$S(\boldsymbol{o}) = \sum_{U_j} p(\boldsymbol{o} \in U_j) \sum_{V_i} p(\boldsymbol{o} \in V_i) p(V_i|U_j). \tag{11.22}$$

Thus the contextual outlier problem is transformed into outlier detection using mixture models.

11.6.2 Modeling normal behavior with respect to contexts

In some applications, it is inconvenient or infeasible to clearly partition the data into contexts. For example, consider the situation where an online store records customer browsing behavior in a search log. For each customer, the data log contains the sequence of products searched for and browsed by the customer. The store is interested in contextual outlier behavior, such as if a customer suddenly purchased

a product that is unrelated to those she recently browsed. However, in this application, contexts cannot be easily specified because it is unclear how many products browsed earlier should be considered as the context, and this number will likely differ for each product.

This second category of contextual outlier detection methods models the normal behavior with respect to contexts. Using a training data set, such a method trains a model that predicts the expected behavior attribute values with respect to the contextual attribute values. To determine whether a data object is a contextual outlier, we can then apply the model to the contextual attributes of the object. If the behavior attribute values of the object significantly deviate from the values predicted by the model, then the object can be declared a contextual outlier.

By using a prediction model that links the contexts and behavior, these methods avoid the explicit identification of specific contexts. A number of classification and prediction techniques can be used to build such models such as regression, Markov models, and finite state automaton. Interested readers are referred to Chapters 6 and 7 on classification and the bibliographic notes for further details (Section 11.10).

In summary, contextual outlier detection enhances conventional outlier detection by considering contexts, which are important in many applications. We may be able to detect outliers that cannot be detected otherwise. Consider a credit card user whose income level is low but whose expenditure patterns are similar to those of millionaires. This user can be detected as a contextual outlier if the income level is used to define context. Such a user may not be detected as an outlier without contextual information because she does share expenditure patterns with many millionaires. Considering contexts in outlier detection can also help to avoid false alarms. Without considering the context, a millionaire's purchase transaction may be falsely detected as an outlier if the majority of customers in the training set are not millionaires. This can be corrected by incorporating contextual information in outlier detection.

11.6.3 Mining collective outliers

A group of data objects forms a **collective outlier** if the objects as a whole deviate significantly from the entire data set, even though each individual object in the group may not be an outlier (Section 11.1). To detect collective outliers, we have to examine the *structure* of the data set, that is, the relationships between multiple data objects. This makes the problem more difficult than conventional and contextual outlier detection.

"How can we explore the data set structure?" This typically depends on the nature of the data. For outlier detection in temporal data (e.g., time series and sequences), we explore the structures formed by time, which occur in segments of the time series or subsequences. To detect collective outliers in spatial data, we explore local areas. Similarly, in graph and network data, we explore subgraphs. Each of these structures is inherent to its respective data type.

Contextual outlier detection and collective outlier detection are similar in that they both explore structures. In contextual outlier detection, the structures are the contexts, as specified by the contextual attributes explicitly. The critical difference in collective outlier detection is that the structures are often not explicitly defined and have to be discovered as part of the outlier detection process.

As with contextual outlier detection, collective outlier detection methods can also be divided into two categories. The first category consists of methods that reduce the problem to conventional outlier detection. Its strategy is to identify *structure units*, treat each structure unit (e.g., a subsequence, a time-series segment, a local area, or a subgraph) as a data object, and extract features. The problem of

collective outlier detection is thus transformed into outlier detection on the set of "structured objects" constructed as such using the extracted features. A structure unit, which represents a group of objects in the original data set, is a collective outlier if the structure unit deviates significantly from the expected trend in the space of the extracted features.

Example 11.27. Collective outlier detection on graph data. Let's see how we can detect collective outliers in a store's online social network of customers. Suppose we treat the social network as an unlabeled graph. We then treat each possible subgraph of the network as a structure unit. For each subgraph, S, let $|S|$ be the number of vertices in S, and $freq(S)$ be the frequency of S in the network. That is, $freq(S)$ is the number of different subgraphs in the network that are isomorphic to S. We can use these two features to detect outlier subgraphs. An *outlier subgraph* is a collective outlier that contains multiple vertices.

In general, a small subgraph (e.g., a single vertex or a pair of vertices connected by an edge or a dense subgraph) is expected to be frequent, and a large subgraph is expected to be infrequent. Using the preceding simple method, we can detect small subgraphs that are of very low frequency or large subgraphs that are surprisingly frequent. These are outlier structures in the social network. □

Predefining the structure units for collective outlier detection can be difficult or impossible. Consequently, the second category of methods models the expected behavior of structure units directly. For example, to detect collective outliers in temporal sequences, one method is to learn a Markov model from the sequences. A subsequence can then be declared as a collective outlier if it significantly deviates from the model.

In summary, collective outlier detection is subtle due to the challenge of exploring the structures in data. The exploration typically uses heuristics and thus may be application-dependent. The computational cost is often high due to the sophisticated mining process. While highly useful in practice, collective outlier detection remains a challenging direction that calls for further research and development.

11.7 Outlier detection in high-dimensional data

In some applications, we may need to detect outliers in high-dimensional data. The curse of dimensionality poses huge challenges for effective outlier detection. As the dimensionality increases, the distance between objects may be heavily dominated by noise. That is, the distance and similarity between two points in a high-dimensional space may not reflect the real relationship between the points. Consequently, conventional outlier detection methods, which mainly use proximity or density to identify outliers, deteriorate as dimensionality increases.

Ideally, outlier detection methods for high-dimensional data should meet the challenges that follow.

- **Interpretation of outliers:** They should be able to not only detect outliers but also provide an interpretation of the outliers. Because many features (or dimensions) are involved in a high-dimensional data set, detecting outliers without providing any interpretation as to why they are outliers is not very useful. The interpretation of outliers may come from, for example, specific subspaces that manifest the outliers or an assessment regarding the "outlier-ness" of the objects. Such interpretation can help users understand the possible meaning and significance of the outliers.

- **Data sparsity:** The methods should be capable of handling sparsity in high-dimensional spaces. The distance between objects becomes heavily dominated by noise as the dimensionality increases. Therefore data in high-dimensional spaces are often sparse.
- **Data subspaces:** They should model outliers appropriately, for example, adaptive to the subspaces signifying the outliers and capturing the local behavior of data. Using a fixed-distance threshold against all subspaces to detect outliers is not a good idea because the distance between two objects monotonically increases as the dimensionality increases.
- **Scalability with respect to dimensionality:** As the dimensionality increases, the number of subspaces increases exponentially. An exhaustive combinatorial exploration of the search space, which contains all possible subspaces, is not a scalable choice.

Outlier detection methods for high-dimensional data can be divided into the following approaches. These include extending conventional outlier detection (Section 11.7.1), finding outliers in subspaces (Section 11.7.2), outlier detection ensemble (Section 11.7.3), deep learning–based approaches (Section 11.7.4) and modeling high-dimensional outliers (Section 11.7.5).

11.7.1 Extending conventional outlier detection

One approach for outlier detection in high-dimensional data extends conventional outlier detection methods. It uses the conventional proximity-based models of outliers. However, to overcome the deterioration of proximity measures in high-dimensional spaces, it uses alternative measures or constructs subspaces and detects outliers there.

The **HilOut** algorithm is an example of this approach. HilOut finds distance-based outliers, but uses the ranks of distance instead of the absolute distance in outlier detection. Specifically, for each object, $\boldsymbol{o}$, HilOut finds the k-nearest neighbors of $\boldsymbol{o}$, denoted by $nn_1(\boldsymbol{o}), \ldots, nn_k(\boldsymbol{o})$, where k is an application-dependent parameter. The weight of object $\boldsymbol{o}$ is defined as

$$w(\boldsymbol{o}) = \sum_{i=1}^{k} dist(\boldsymbol{o}, nn_i(\boldsymbol{o})). \tag{11.23}$$

All objects are ranked in weight-descending order. The top-l objects in weight are output as outliers, where l is another user-specified parameter.

Computing the k-nearest neighbors for every object is costly and does not scale up when the dimensionality is high and the database is large. To address the scalability issue, HilOut employs space-filling curves to achieve an approximation algorithm, which is scalable in both running time and space with respect to database size and dimensionality.

While some methods like HilOut detect outliers in the full space despite the high dimensionality, other methods reduce the high-dimensional outlier detection problem to a lower-dimensional one by dimensionality reduction (Chapter 2). The idea is to reduce the high-dimensional space to a lower-dimensional space where normal instances can still be distinguished from outliers. If such a lower-dimensional space can be found, then conventional outlier detection methods can be applied. In principle, the matrix factorization based approaches introduced in Section 11.4 can be used to find such lower-dimensional space. For example, the rows of the right matrix $\mathbf{G}$ in Fig. 11.10 provide the representation of the input data tuples in the lower-dimensional space.

To reduce the dimensionality, general feature selection and extraction methods may be used or extended for outlier detection. For example, principal components analysis (PCA) can be used to extract a lower-dimensional space. Heuristically, the principal components with *low* variance are preferred because, on such dimensions, normal objects are likely close to each other and outliers often deviate from the majority. Notice that this is different from the case when PCA is used as a general-purposed dimensionality reduction tool, where the principal components with *high* variance are favored.

By extending conventional outlier detection methods, we can reuse much of the experience gained from research in the field. These new methods, however, are limited. First, they cannot detect outliers with respect to subspaces and thus have limited interpretability. Second, dimensionality reduction is feasible only if there exists a lower-dimensional space where normal objects and outliers are well separated. This assumption may not hold true.

11.7.2 Finding outliers in subspaces

Another approach for outlier detection in high-dimensional data is to search for outliers in various subspaces. A unique advantage is that, if an object is found to be an outlier in a subspace of much lower dimensionality, the subspace provides critical information for interpreting *why* and *to what extent* the object is an outlier. This insight is highly valuable in applications with high-dimensional data due to the overwhelming number of dimensions.

Example 11.28. Outliers in subspaces. As a customer-relationship manager at an electronics store, you are interested in finding outlier customers. The store maintains an extensive customer information database, which contains many attributes and the transaction history of customers. The database is high dimensional.

Suppose you find that a customer, Alice, is an outlier in a lower-dimensional subspace that contains the dimensions *average_transaction_amount* and *purchase_frequency*, such that her average transaction amount is substantially larger than the majority of the customers, and her purchase frequency is dramatically lower. The subspace itself speaks for why and to what extent Alice is an outlier. Using this information, you strategically decide to approach Alice by suggesting options that could improve her purchase frequency at the store. □

"How can we detect outliers in subspaces?" We use a *grid-based subspace outlier detection method* to illustrate. The major ideas are as follows. We consider projections of the data onto various subspaces. If, in a subspace, we find an area that has a density that is much lower than average, then the area may contain outliers. To find such projections, we first discretize the data into a grid in an equal-depth way. That is, each dimension is partitioned into ϕ equal-depth ranges, where each range contains a fraction, f, of the objects $\left(f = \frac{1}{\phi}\right)$. Equal-depth partitioning is used because data along different dimensions may have different localities. An equal-width partitioning of the space may not be able to reflect such differences in locality.

Next, we search for regions defined by ranges in subspaces that are significantly sparse. To quantify what we mean by "significantly sparse," let's consider a k-dimensional cube formed by k ranges on k dimensions. Suppose the data set contains n objects. If the objects are independently distributed, the expected number of objects falling into a k-dimensional region is $\left(\frac{1}{\phi}\right)^k n = f^k n$. The standard deviation of the number of points in a k-dimensional region is $\sqrt{f^k(1 - f^k)n}$. Suppose a specific k-dimensional

cube C has $n(C)$ objects. We can define the **sparsity coefficient** of C as

$$S(C) = \frac{n(C) - f^k n}{\sqrt{f^k(1 - f^k)n}}. \tag{11.24}$$

If $S(C) < 0$, then C contains fewer objects than expected. The smaller the value of $S(C)$ (i.e., the more negative), the sparser C is and the more likely the objects in C are outliers in the subspace.

By assuming $S(C)$ follows a normal distribution, we can use normal distribution tables to determine the significance level for an object that deviates dramatically from the average for a priori assumption of the data following a uniform distribution. In general, the assumption of uniform distribution does not hold. However, the sparsity coefficient still provides an intuitive measure of the "outlier-ness" of a region.

To find cubes of significantly small sparsity coefficient values, a brute-force approach is to search every cube in every possible subspace. The cost of this, however, is immediately exponential. An *evolutionary search* can be conducted, which improves efficiency at the expense of accuracy. For details, please refer to the bibliographic notes (Section 11.10). The objects contained by cubes of very small sparsity coefficient values are output as outliers. In addition, *genetic algorithm* has been found to be effective in searching for desirable cubes.

In summary, searching for outliers in subspaces is advantageous in that the outliers found tend to be better understood, owing to the context provided by the subspaces.[6] Challenges include making the search efficient and scalable.

11.7.3 Outlier detection ensemble

Another effective way to detect outliers in high-dimensional data is via *ensemble*. Its general procedure is as follows and a pictorial illustration is given in Fig. 11.22. Given an input data set $\mathbf{X}$ (the left of Fig. 11.22), with n data tuples (rows) in d-dimensional space (columns), the ensemble methods first create K base detectors (the middle of Fig. 11.22). For each base detector, we first represent the input data set in a *random subspace* $\mathbf{X}_i (i = 1, ..., K)$ whose dimensionality d' is much smaller than the original feature dimensionality (i.e., $d' \ll d$), and then use an off-the-shelf outlier detection method (e.g., LOF) to assign each data tuple with an outlier-ness score. We represent the outlier-ness scores of all n data tuples in the form of a vector $\mathbf{y}_i$ of length n $(i = 1, ..., K)$. After that, we aggregate the detection results from the K base detectors to obtain the overall outlier-ness scores for all input data tuples in the form of another vector $\mathbf{y}$ of length n (the right of Fig. 11.22). Finally, we flag data tuples with the highest overall outlier-ness scores as outliers.

There are two key issues in outlier detection ensemble, including (1) how to represent the input data in a random subspace and (2) how to aggregate the detection results of base detectors. For (1), there are two commonly used methods, including *feature bagging* and *rotated bagging*. For feature bagging, we randomly select a few original features to form the representation of the input data in a random subspace; and for rotated bagging, we first generate a random subspace of dimensionality $d'(d' \ll d)$ and then project the input data into this subspace. Conceptually, if we represent the input

[6] For this reason, we can see that subspace-based outlier detection and contextual outlier detection are closely related with each other.

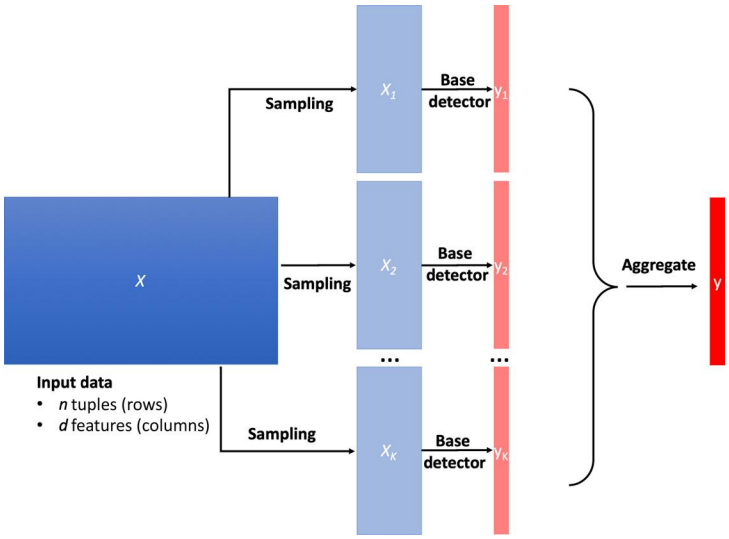


FIGURE 11.22

A pictorial illustration of using ensemble methods to detect outliers in high-dimensional data. (Left) The input data set with n data tuples (rows) in d-dimensional space (columns). (Middle) K base detectors. (Right) The aggregated detection results.

data set as an $n \times d$ data matrix $\mathbf{X}$, feature bagging selects a few $d'(d' \ll d)$ actual columns of $\mathbf{X}$ as the representation of the input data set in a random subspace, whereas rotated bagging first generates d' mutually orthonormal vectors and then projects the input data matrix $\mathbf{X}$ onto the subspace spanned by these d' vectors.

In order to aggregate the detection results from the base detectors, we can either use *mean aggregation* where the overall outlier-ness score of a given data tuple is the average of its outlier-ness scores from K base detectors, or *max aggregation* where the overall outlier-ness score of a given data tuple is the maximum of its outlier-ness scores from K base detectors. For both aggregation methods, it is important to normalize (e.g., min-max normalization, z-score normalization) the outlier-ness scores of the base detectors *before* aggregation, so that the outlier-ness score of a base detector will not overwhelm the scores of other base detectors.

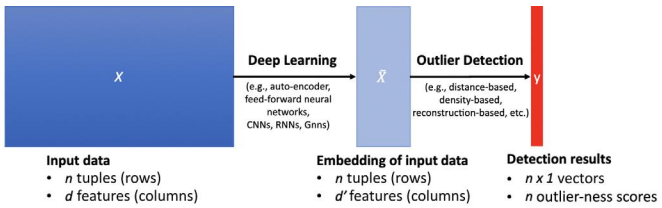
11.7.4 Taming high dimensionality by deep learning

In the context of outlier detection in high-dimensional data, deep learning–based approaches offer two appealing advantages. First, deep learning methods (e.g., autoencoder, feed-forward neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), graphical neural networks (GNNs)) produce vector representation (i.e., *embedding*) of the input data tuples with a much smaller number of dimensions than the original ones and thus naturally alleviate the high-dimesionality challenge. Second, owing to deep learning's strong ability to learning semantically meaningful representation, the produced embedding often captures the complicate (e.g., nonlinear) interaction between

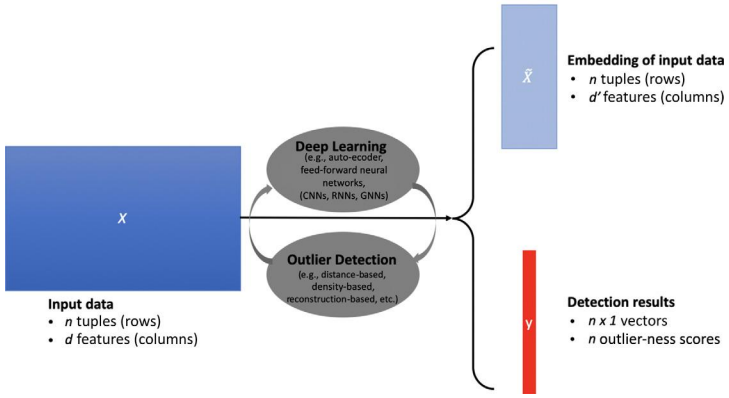
different input futures and thus is capable of detecting outliers that might be overlooked by the alternative methods (e.g., linear matrix factorization–based approaches).

Generally speaking, there are two basic strategies to leverage deep learning for high-dimensional outlier detection, which we will introduce next. Fig. 11.23 presents an illustration.

The first strategy uses the deep learning methods as a *preprocessing* step. For example, if we feed the input data set into an autoencoder, the output from the encoder produces the embedding of the input data tuples in a much lower-dimensional space. Then, with such embedding as the input, we can use outlier detection methods (e.g., proximity-based approaches, reconstruction-based approaches) to detect outlying tuples. The advantage of this type of strategy lies in its simplicity. In principle, we can use a variety of deep learning models to produce embedding of the input data in a lower-dimensional space, such as feed-forward neural networks for spatial data, convolutional neural networks for grid data, recurrent neural networks for sequential data, and graph neural networks for graph data. The learned embedding can in turn be fed into a variety of off-the-shelf outlier detection methods, without the need of retraining or modifications. However, this strategy separates the embedding learning stage and outlier detection stage and thus might lead to suboptimal detection results. For example, the outliers in the learned embedding space might be mixed with that of the normal tuples and thus are missed by the outlier detection methods.



(a) Deep learning as a pre-processing step



(b) Integrating deep learning with outlier detection

FIGURE 11.23

An illustration of two basic strategies of using deep learning for high-dimensional outlier detection.

The second strategy aims to integrate the embedding learning and outlier detection together. In other words, it tries to find embedding of the input data tuples that is tailored for spotting the outliers and simultaneously produce the embedding and detection results. Compared with the first strategy, it often leads to a higher detection accuracy. The general idea of methods using this strategy is to replace certain component of an existing outlier detection method by a deep learning model. Let us take one-class SVM as an example. Recall that in the traditional one-class SVM (Section 11.5.2), it seeks to find a max-margin hyperplane in the transformed high-dimensional space (i.e., RKHS) that separates the normal data tuples from the origin which represents the outlier. To this end, it solves an optimization problem that involves a component of $w^T \phi(x)$ representing the output of the hyperplane, where w is the weight vector and $\phi(x)$ in the feature vector of the input data x in the RKHS. In the **One-Class Neural Network (OC-NN)**, it replaces the hyperplane output (i.e., $w^T \phi(x)$) by a feed-forward neural network, whose output layer produces an outlier-ness score, and the last hidden layer produces the embedding of the input data. By combining the embedding learning and outlier detection together, it was found that **OC-NN** leads to better performance than the traditional one-class SVM.

Another example is **Deviation Networks (DevNet)**. Recall that for univariate outlier detection (Section 11.2.1), a simple yet effective outlier detection method is Grubb's test based on z-score: $z = \frac{|x-\mu|}{\sigma}$, where x is the input feature value, and μ and σ are the sample mean and the sample standard deviation, respectively. For high-dimensional data, **DevNet** replaces the raw feature value x by its outliner-ness score $f(x, \Theta)$. The outliner-ness score $f(x, \Theta)$ is produced by a deep learning model (e.g., a feed-forward neural network with a linear output layer) with x as the input and Θ being the model parameters of the deep learning model. In addition, **DevNet** replaces μ and σ by the sample mean and the sample standard deviation of the outlier-ness scores of the normal tuples, respectively.[7] DevNet was found to obtain significant detection accuracy improvement over alternative methods.

11.7.5 Modeling high-dimensional outliers

Another alternative approach for outlier detection methods in high-dimensional data tries to develop new models for high-dimensional outliers directly. Such models typically avoid proximity measures and instead adopt new heuristics to detect outliers, which do not deteriorate in high-dimensional data.

Let's examine *angle-based outlier detection* (ABOD) as an example.

Example 11.29. Angle-based outliers. Fig. 11.24 contains a set of points forming a cluster, with the exception of $\boldsymbol{c}$, which is an outlier. For each point $\boldsymbol{o}$, we examine the angle $\angle \boldsymbol{xoy}$ for every pair of points $\boldsymbol{x}$, $\boldsymbol{y}$ such that $\boldsymbol{x} \neq \boldsymbol{o}$, $\boldsymbol{y} \neq \boldsymbol{o}$. The figure shows angle $\angle \boldsymbol{dae}$ as an example.

Note that for a point in the center of a cluster (e.g., $\boldsymbol{a}$), the angles formed as such differ widely. For a point that is at the border of a cluster (e.g., $\boldsymbol{b}$), the angle variation is smaller. For a point that is an outlier (e.g., $\boldsymbol{c}$), the angle variable is substantially smaller. This observation suggests that we can use the variance of angles for a point to determine whether a point is an outlier. □

We can combine angles and distance to model outliers. Mathematically, for each point $\boldsymbol{o}$, we use the distance-weighted angle variance as the outlier-ness score. That is, given a set of points, D, for a point,

[7] Another subtle change DevNet made is to remove the absolute value sign in the z-score definition. In other words, it only looks at the upper-tail of the outlier-ness scores produced by the deep learning model. The larger the outlier-ness score, the more likely the given tuple is an outlier.

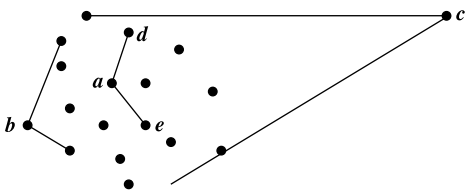


FIGURE 11.24

Angle-based outliers.

$o \in D$, we define the **angle-based outlier factor** (ABOF) as

$$ABOF(o) = \text{VAR}_{x,y \in D, x \neq o, y \neq o} \frac{\langle \overrightarrow{ox}, \overrightarrow{oy} \rangle}{dist(o,x)^2 dist(o,y)^2}, \tag{11.25}$$

where $\langle , \rangle$ is the scalar product (i.e., dot product) operator, VAR is the variance, and $dist(,)$ is a norm distance.

Clearly, the farther away a point is from clusters and the smaller the variance of the angles of a point, the smaller the ABOF. The ABOD computes the ABOF for each point and outputs a list of the points in the data set in ABOF-ascending order.

Computing the exact ABOF for every point in a database is costly, requiring a time complexity of $O(n^3)$, where n is the number of points in the database. Obviously, this exact algorithm does not scale up for large data sets. Approximation methods have been developed to speed up the computation. The angle-based outlier detection idea has been generalized to handle arbitrary data types. For additional details, see the bibliographic notes (Section 11.10).

Developing native models for high-dimensional outliers can lead to effective methods. However, finding good heuristics for detecting high-dimensional outliers is difficult. Efficiency and scalability on large and high-dimensional data sets are major challenges.

11.8 Summary

- Assume that a given statistical process is used to generate a set of data objects. An **outlier** is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism.
- **Types of outliers** include global outliers, contextual outliers, and collective outliers. An object may be more than one type of outlier.
- **Global outliers** are the simplest form of outlier and the easiest to detect. A **contextual outlier** deviates significantly with respect to a specific context of the object (e.g., a Toronto temperature value of 28 °C is an outlier if it occurs in the context of winter). A subset of data objects forms a **collective outlier** if the objects as a whole deviate significantly from the entire data set, even though the individual data objects may not be outliers. Collective outlier detection requires background information to model the relationships among objects to find outlier groups.

- **Challenges** in outlier detection include finding appropriate data models, the dependence of outlier detection systems on the application involved, finding ways to distinguish outliers from noise, and providing justification for identifying outliers as such.
- Outlier detection methods can be **categorized** according to whether the sample of data for analysis is given with expert-provided labels that can be used to build an outlier detection model. In this case, the detection methods are *supervised, semisupervised*, or *unsupervised*. Alternatively, outlier detection methods may be organized according to their assumptions regarding normal objects vs. outliers. This categorization includes *statistical* methods, *proximity-based* methods, and *reconstruction-based* methods.
- **Statistical outlier detection methods** (or **model-based methods**) assume that the normal data objects follow a statistical model, where data not following the model are considered outliers. Such methods may be *parametric* (they assume that the data are generated by a parametric distribution) or *nonparametric* (they learn a model for the data rather than assuming one a priori). Parametric methods for multivariate data may employ the Mahalanobis distance, the χ^2 statistic, or a mixture of multiple parametric models. Histograms and kernel density estimation are examples of nonparametric methods.
- **Proximity-based outlier detection methods** assume that an object is an outlier if the proximity of the object to its nearest neighbors significantly deviates from the proximity of most of the other objects to their neighbors in the same data set. *Distance-based outlier detection methods* consult the *neighborhood* of an object, defined by a given radius. An object is an outlier if its neighborhood does not have enough other points. In *density-based outlier detection methods*, an object is an outlier if its density is relatively much lower than that of its neighbors.
- **Clustering-based outlier detection methods** assume that the normal data objects belong to large and dense clusters, whereas outliers belong to small or sparse clusters or do not belong to any clusters.
- **Classification-based outlier detection methods** often use a one-class model. That is, a classifier is built to describe only the normal class. Any samples that do not belong to the normal class are regarded as outliers.
- **Contextual outlier detection** and **collective outlier detection** explore structures in the data. In contextual outlier detection, the structures are defined as contexts using contextual attributes. In collective outlier detection, the structures are implicit and are explored as part of the mining process. To detect such outliers, one approach transforms the problem into one of conventional outlier detection. Another approach models the structures directly.
- **Outlier detection methods for high-dimensional data** can be divided into five main approaches, including extending conventional outlier detection, finding outliers in subspaces, outlier detection ensemble, deep learning–based approaches, and modeling high-dimensional outliers.

11.9 Exercises

11.1. Give an application example where global outliers, contextual outliers, and collective outliers are all interesting. What are the attributes, and what are the contextual and behavioral attributes? How is the relationship among objects modeled in collective outlier detection?

11.2. Give an application example of where the border between normal objects and outliers is often unclear, so that the degree to which an object is an outlier has to be well estimated.

11.3. Adapt a simple semisupervised method for outlier detection. Discuss the scenario where you have (a) only some labeled examples of normal objects and (b) only some labeled examples of outliers.

11.4. Using an equal-depth histogram, design a way to assign an object an outlier-ness score.

11.5. Consider the nested loop approach to mining distance-based outliers (Fig. 11.6). Suppose the objects in a data set are arranged randomly; that is, each object has the same probability to appear in a position. Show that when the number of outlier objects is small with respect to the total number of objects in the whole data set, the expected number of distance calculations is linear to the number of objects.

11.6. In the density-based outlier detection method of Section 11.3.2, the definition of local reachability density has a potential problem: $lrd_k(o) = \infty$ may occur. Explain why this may occur and propose a fix to the issue.

11.7. Because clusters may form a hierarchy, outliers may belong to different granularity levels. Propose a clustering-based outlier detection method that can find outliers at different levels.

11.8. In outlier detection by semisupervised learning, what is the advantage of using objects without labels in the training data set?

11.9. Given a user-community bipartite graph, where the nodes are users and communities, and links indicate the membership between users and communities. We can represent this bipartite graph by its adjacency matrix $\mathbf{A}$, where $\mathbf{A}(i, j) = 1$ means user i belongs to community j; and $\mathbf{A}(i, j) = 0$ otherwise. We further approximate the adjacency matrix $\mathbf{A}$ by the multiplication of two low-rank matrices, that is, $\mathbf{A} \approx \mathbf{FG}$, where $\mathbf{F}$ and $\mathbf{G}$ are two low-rank matrices. Describe how you can leverage the above low-rank approximation result to detect (a) outlying users and (b) outlying user-community memberships, respectively.

11.10. Describe a method to integrate support vector data description (SVDD) and deep learning for outlier detection.

11.11. To understand why angle-based outlier detection is a heuristic method, give an example where it does not work well. Can you come up with a method to overcome this issue?

11.10 Bibliographic notes

Hawkins [Haw80] defined outliers from a statistics angle. For surveys or tutorials on the subject of outlier and anomaly detection, see Chandola, Banerjee, and Kumar [CBK09]; Hodge and Austin [HA04]; Agyemang, Barker, and Alhajj [ABA06]; Markou and Singh [MS03a,MS03b]; Patcha and Park [PP07]; Beckman and Cook [BC83]; Ben-Gal [BG05]; and Bakar, Mohemad, Ahmad, and Deris [BMAD06]. For outlier detection on temporal data, see Gupta, Gao, Aggarwal, and Han [GGAH13]. For anomaly detection on graph data, see Akoglu, Tong, and Koutra [ATK15]. Song et al. [SWJR07] proposed the notion of conditional anomaly and contextual outlier detection. For a comprehensive textbook on outlier detection, see Aggarwal [Agg15c].

Fujimaki, Yairi, and Machida [FYM05] presented an example of semisupervised outlier detection using a set of labeled "normal" objects. For an example of semisupervised outlier detection using labeled outliers, see Dasgupta and Majumdar [DM02]. For outlier detection with active learning, see

He and Carbonell [HC08]; Prateek and Ashish [JK09]; He, Liu, and Richard [HLL08]; Hospedales, Gong, and Xiang [HGX11]; and Zhou and He [ZHCD15].

Shewhart [She31] assumed that most objects follow a Gaussian distribution and used 3σ as the threshold for identifying outliers, where σ is the standard deviation. Boxplots are used to detect and visualize outliers in various applications such as medical data (Horn, Feng, Li, and Pesce [HFLP01]). Grubb's test was described by Grubbs [Gru69], Stefansky [Ste72], and Anscombe and Guttman [AG60]. Laurikkala, Juhola, and Kentala [LJK00] and Aggarwal and Yu [AY01] extended Grubb's test to detect multivariate outliers. Use of the χ^2 statistic to detect multivariate outliers was studied by Ye and Chen [YC01].

Agarwal [Aga06] used Gaussian mixture models to capture "normal data." Abraham and Box [AB79] assumed that outliers are generated by a normal distribution with a substantially larger variance. Eskin [Esk00] used the EM algorithm to learn mixture models for normal data and outliers.

Histogram-based outlier detection methods are popular in the application domain of intrusion detection (Eskin [Esk00] and Eskin et al. [EAP+02]) and fault detection (Fawcett and Provost [FP97]).

The notion of distance-based outliers was developed by Knorr and Ng [KN97]. The index-based, nested loop–based, and grid-based approaches were explored (Knorr and Ng [KN98] and Knorr, Ng, and Tucakov [KNT00]) to speed up distance-based outlier detection. Bay and Schwabacher [BS03] and Jin, Tung, and Han [JTH01] pointed out that the CPU runtime of the nested loop method is often scalable with respect to database size. Tao, Xiao, and Zhou [TXZ06] presented an algorithm that finds all distance-based outliers by scanning the database three times with fixed main memory. For larger memory size, they proposed a method that uses only one or two scans.

The notion of density-based outliers was first developed by Breunig, Kriegel, Ng, and Sander [BKNS00]. Various methods proposed with the theme of density-based outlier detection include Jin, Tung, and Han [JTH01]; Jin, Tung, Han, and Wang [JTHW06]; and Papadimitriou et al. [PKGF03]. The variations differ in how they estimate density.

SVD and related techniques were used in [IK04] to detect anomalies in computer systems. Incremental SVD was developed by Papadimitriou, Sun, and Faloutsos [PSF05] to find anomalies in coevolving time series data. Principal component analysis (PCA) was used to detect abnormal traffic in [BSM09]. For example-based matrix factorization, see Mahoney and Drineas [MD09] and Tong et al. [TPS+08]. For nonnegative residual matrix factorization, see Tong and Lin [TL11]. Xu, Constantine, and Sujay proposed robust PCA for outlier detection [XCS12]. Pattern-compression–based outlier detection was first developed by Vreeken, Leeuwen, and Siebes [VvLS11]. Akoglu, Tong, Vreeken, and Faloutsos [ATVF12] further improved the method by building multiple codetables.

The bootstrap method discussed in Example 11.21 was developed by Barbara et al. [BLC+03]. The FindCBOLF algorithm was given by He, Xu, and Deng [HXD03]. For the use of fixed-width clustering in outlier detection methods, see Eskin et al. [EAP+02]; Mahoney and Chan [MC03]; and He, Xu, and Deng [HXD03]. Barbara, Wu, and Jajodia [BWJ01] used multiclass classification in network intrusion detection.

Song et al. [SWJR07] and Fawcett and Provost [FP97] presented a method to reduce the problem of contextual outlier detection to one of conventional outlier detection. Yi et al. [YSJ+00] used regression techniques to detect contextual outliers in coevolving sequences. The idea in Example 11.26 for collective outlier detection on graph data is based on Noble and Cook [NC03].

The HilOut algorithm was proposed by Angiulli and Pizzuti [AP05]. Aggarwal and Yu [AY01] developed the sparsity coefficient–based subspace outlier detection method. Kriegel, Schubert, and Zimek [KSZ08] proposed angle-based outlier detection.

Zhou and Paffenroth introduced deep autoencoder for outlier detection [ZP17]. Chalapathy, Menon, and Chawla proposed to integrate one-class SVM with neural networks for effective outlier detection [CMC18]. Pang, Shen, and Hengel developed deviation networks for deep outlier detection. For surveys on deep learning–based outlier detection, see Chalapathy and Chawla [CC19] and Pang, Shen, Cao, and Hengel [PSCH20].

There are numerous applications of outlier and anomaly detection, ranging from finance [NHW+11, ZZY+17], healthcare [vCPM+16], accounting [MBA+09], intrusion detection [ZZ06], multiarmed bandit [ZWW17,BH20], to misinformation [ZG20].