12 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 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/video network surveillance, 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 this chapter, we study outlier detection techniques. Section 12.1 defines the different types of outliers. Section 12.2 presents 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 12.3), proximity-based (Section 12.4), clustering-based (Section 12.5), and classification-based (Section 12.6). In addition, you will learn about mining contextual and collective outliers (Section 12.7) and outlier detection in high-dimensional data (Section 12.8).

12.1 Outliers and Outlier Analysis

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.

12.1.1 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 12.1 Outliers. In Figure 12.1, most objects follow a roughly 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 3, 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 "noise 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 mechanisms as the rest of the data. Therefore, in outlier detection, it is important to

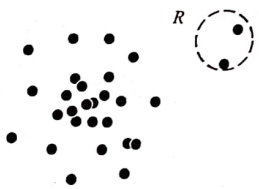


Figure 12.1 The objects in region R are outliers.

justify *why* the outliers detected are generated by some other mechanisms. This is often achieved by making various assumptions on the rest of the data and showing that the outliers detected 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.

12.1.2 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 12.2 Global outliers. Consider the points in Figure 12.1 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 are proposed, 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,

...e other mechanisms. This is often
...f the data and showing that the

...ving data sets. For example,
...oming, novelty detection
...ics may initially appear
...hare some similarity
...etween the two is
...ly incorporated
...ed as outliers

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 12.4.3.

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 12.3 ***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 and/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 12.4.

Collective Outliers

Suppose you are a supply-chain manager of *AllElectronics*. 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 12.4 **Collective outliers.** In Figure 12.2, 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.

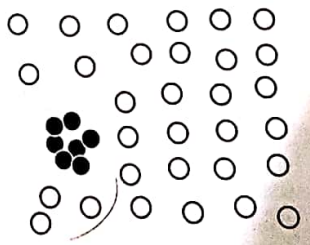


Figure 12.2 The black objects form a collective outlier.

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.

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. Context 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.

12.1.3 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" of the object.

- **Application-specific outlier detection.** Technically, choosing the similarity/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 subject to larger fluctuations, and consequently a substantially larger deviation is needed to justify an outlier. Outlier detection's 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 noise point, and an outlier detection method may mistakenly identify a noise point as an outlier.

- **Understandability.** 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.

The rest of this chapter discusses approaches to outlier detection.

12.2 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 versus outliers.

12.2.1 Supervised, Semi-Supervised, and Unsupervised Methods

If expert-labeled examples of normal and/or outlier objects can be obtained, they can be used to build outlier detection models. The methods used can be divided into supervised methods, semi-supervised 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 8 and 9). 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 versus 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 8.6.5) 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 classifiers built as such. 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 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 Figure 12.2, 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 cannot 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 are detected 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. The latest 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 12.4 and 12.5 on proximity-based and clustering-based methods, respectively.

Semi-Supervised 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/or outlier objects are labeled, but most of the data are unlabeled. Semi-supervised outlier detection methods were developed to tackle such scenarios.

Semi-supervised outlier detection methods can be regarded as applications of semi-supervised learning methods (Section 9.7.2). 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, semi-supervised 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 semi-supervised methods, interested readers are referred to the bibliographic notes at the end of this chapter (Section 12.11).

12.2.2 Statistical Methods, Proximity-Based Methods, and Clustering-Based Methods

As discussed in Section 12.1, outlier detection methods make assumptions about outliers versus 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 clustering-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 12.5 **Detecting outliers using a statistical (Gaussian) model.** In Figure 12.1, 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 12.3.

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 12.6 **Detecting outliers using proximity.** Consider the objects in Figure 12.1 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 12.4.

Clustering-Based Methods

Clustering-based 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.

Example 12.7 **Detecting outliers using clustering.** In Figure 12.1, there are two clusters. Cluster C_1 contains all the points in the data set except for those in region R. Cluster C_2 is small, containing just two points in R. Cluster C_1 is large in comparison to C_2. Therefore, a clustering-based method asserts that the two objects in R are outliers.

There are many clustering methods, as discussed in Chapters 10 and 11. Therefore, there are many clustering-based outlier detection methods as well. Clustering is an expensive data mining operation. A straightforward adaptation of a clustering method for outlier detection can be very costly, and thus does not scale up well for large data sets. Clustering-based outlier detection methods are discussed in detail in Section 12.5.

12.3 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 (a generative model). Consequently, normal objects occur in 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 parameter 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. 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 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.

12.3.1 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 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 12.8 Univariate outlier detection using maximum likelihood. 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*

$$\ln \mathcal{L}(\mu, \sigma^2) = \sum_{i=1}^{n} \ln f(x_i | (\mu, \sigma^2)) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2, \quad (12.1)$$

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*:

$$\hat{\mu} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (12.2)$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2. \quad (12.3)$$

In this example, we have

$$\hat{\mu} = \frac{24.0 + 28.9 + 28.9 + 29.0 + 29.1 + 29.1 + 29.2 + 29.2 + 29.3 + 29.4}{10} = 28.61$$

$$\begin{aligned}\hat{\sigma}^2 = ((24.1 - 28.61)^2 &+ (28.9 - 28.61)^2 + (28.9 - 28.61)^2 + (29.0 - 28.61)^2 \\ &+ (29.1 - 28.61)^2 + (29.1 - 28.61)^2 + (29.2 - 28.61)^2 + (29.2 - 28.61)^2 \\ &+ (29.3 - 28.61)^2 + (29.4 - 28.61)^2)/10 \simeq 2.29.\end{aligned}$$

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

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

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{12.9}$$

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, 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 normal data objects from outliers. 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 fully understand. 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 12.8.

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.

12.4 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 the object to most of the other objects 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 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.

objects within a distance r from o_i, the inner loop can be terminated because o_i already violates (Eq. 12.10), and thus is not a $DB(r,\pi)$-outlier. On the other hand, if the inner loop completes for o_i, this means that 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 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.

When mining large data sets where the complete set of objects cannot be held in main memory, the nested loop approach is still costly. Suppose the main memory has m pages for the mining. Instead of conducting the inner loop object by object, in such a case, the outer loop uses $m-1$ pages to hold as many objects as possible and uses the remaining one page to run the inner loop. The inner loop cannot stop until all objects in the $m-1$ pages are identified as not being outliers, which is very unlikely to happen. Correspondingly, it is likely that the algorithm has to incur $O((\frac{n}{b})^2)$ input/output (I/O) cost, where b is the number of objects that can be held in one page.

The major cost in the nested loop method comes from two aspects. First, to check whether an object is an outlier, the nested loop method tests the object against the whole data set. To improve, we need to explore how to determine the outlierness of an object from the neighbors that are close to the object. Second, the nested loop method checks objects one by one. To improve, we should try to group objects according to their proximity, and check the outlierness of objects group by group most of the time. Section 12.4.2 introduces how to implement the preceding ideas.

12.4.2 A Grid-Based Method

CELL is a grid-based method for distance-based outlier detection. In this method, the data space is partitioned into a multidimensional grid, where each cell is a hypercube that has a diagonal of length $\frac{r}{2}$, where r is a distance threshold parameter. In other words, if there are l dimensions, the length of each edge of a cell is $\frac{r}{2\sqrt{l}}$.

Consider a 2-D data set, for example. Figure 12.7 shows part of the grid. The length of each edge of a cell is $\frac{r}{2\sqrt{2}}$.

Consider the cell C in Figure 12.7. The neighboring cells of C can be divided into two groups. The cells immediately next to C constitute the *level-1* cells (labeled "1" in the figure), and the cells one or two cells away from C in any direction constitute the *level-2* cells (labeled "2" in the figure). The two levels of cells have the following properties:

- **Level-1 cell property:** Given any possible point, x of C, and any possible point, y, in a level-1 cell, then $dist(x,y) \leq r$.
- **Level-2 cell property:** Given any possible point, x of C, and any point, y, such that $dist(x,y) \geq r$, then y is in a level-2 cell.

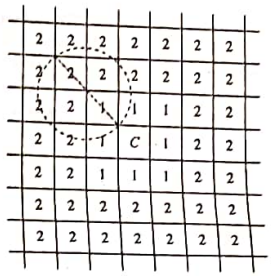


Figure 12.7 Grids in the CELL method.

Let a be the number of objects in cell C, b_1 be the total number of objects in the level-1 cells, and b_2 be the total number of objects in the level-2 cells. We can apply the following rules.

- **Level-1 cell pruning rule:** Based on the level-1 cell property, if $a + b_1 > \lceil \pi n \rceil$, then every object o in C is not a $DB(r,\pi)$-outlier because all those objects in C and the level-1 cells are in the r-neighborhood of o, and there are at least $\lceil \pi n \rceil$ such neighbors.
- **Level-2 cell pruning rule:** Based on the level-2 cell property, if $a + b_1 + b_2 < \lceil \pi n \rceil + 1$, then all objects in C are $DB(r,\pi)$-outliers because each of their r-neighborhoods has less than $\lceil \pi n \rceil$ other objects.

Using the preceding two rules, the CELL method organizes objects into groups using a grid—all objects in a cell form a group. For groups satisfying one of the two rules, we can determine that either all objects in a cell are outliers or nonoutliers, and thus do not need to check those objects one by one. Moreover, to apply the two rules, we need only check a limited number of cells close to a target cell instead of the whole data set.

Using the previous two rules, many objects can be determined as being either nonoutliers or outliers. We only need to check the objects that cannot be pruned using the two rules. Even for such an object, o, we need only compute the distance between o and the objects in the level-2 cells with respect to o. This is because all objects in the level-1 cells have a distance of at most r to o, and all objects not in a level-1 or level-2 cell must have a distance of more than r from o, and thus cannot be in the r-neighborhood of o.

When the data set is very large so that most of the data are stored on disk, the CELL method may incur many random accesses to disk, which is costly. An alternative method was proposed, which uses a very small amount of main memory (around 1% of the data

set) to mine all outliers by scanning the data set three times. First, a sample, S, is created of the given data set, D, using sampling by replacement. Each object in S is considered the centroid of a partition. The objects in D are assigned to the partitions based on distance. The preceding steps are completed in one scan of D. Candidate outliers are identified in a second scan of D. After a third scan, all $DB(r,\pi)$-outliers have been found.

12.4.3 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 12.14 Local proximity-based outliers. Consider the data points in Figure 12.8. There are two clusters: C_1 is dense, and C_2 is sparse. Object 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 o_1 and o_2. Are they outliers? On the one hand, the distance from o_1 and 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, o_1 and o_2 are not distance-based outliers. In fact, if we were to categorize o_1 and o_2 as $DB(r,\pi)$-outliers, we would have to classify all the objects in clusters C_2 as $DB(r,\pi)$-outliers.

On the other hand, o_1 and o_2 can be identified as outliers when they are considered locally with respect to cluster C_1 because o_1 and o_2 deviate significantly from the objects in C_1. Moreover, o_1 and o_2 are also far from the objects in C_2.

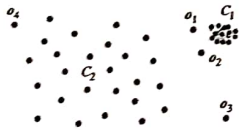


Figure 12.8 Global outliers and local outliers.

To summarize, distance-based outlier detection methods consider certain objects as outliers like o_1 and o_2. Note that the distance between object o_4 and its nearest neighbor is much greater than the distance between o_1 and its nearest neighbor. However, because o_4 is local to cluster C_2 (which is sparse), o_4 is not considered a local outlier.

"How can we formulate the local outliers as illustrated in Example 12.14? The key 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. Given a set of objects, D. The *k-distance* of o, denoted by $dist_k(o)$, is the distance, $dist(o,p)$, between o and another object, $p \in D$, such that

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

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

$$N_k(o) = \{o' | o' \in D, dist(o,o') \leq dist_k(o)\}. \tag{12.11}$$

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

We can use the average distance from the objects in $N_k(o)$ to o as the measure of the local density of o. However, such a straightforward measure has a problem. If o has very close neighbors o' such that $dist(o,o')$ 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, o and o', the *reachability distance* from o' to o is $dist(o \leftarrow o')$ if this $(o,o') > dist_k(o)$, and $dist_k(o)$ otherwise. That is,

$$reachdist_k(o \leftarrow o') = \max\{dist_k(o), dist(o,o')\}. \tag{12.12}$$

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, $reachdist_k(o \leftarrow o') \neq reachdist_k(o' \leftarrow o)$.

Now, we can define the *local reachability density* of an object, o, as

$$lrd_k(o) = \frac{\|N_k(o)\|}{\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)}. \quad (12.13)$$

There is a critical difference between the density measure here for outlier detection and that in density-based clustering (Section 12.5). 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 o as

$$LOF_k(o) = \frac{\sum_{o' \in N_k(o)} \frac{lrd_k(o')}{lrd_k(o)}}{\|N_k(o)\|} = \sum_{o' \in N_k(o)} lrd_k(o') \cdot \sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o). \quad (12.14)$$

In other words, the local outlier factor is the average of the ratio of the local reachability density of o and those of o's k-nearest neighbors. The lower the local reachability density of o (i.e., the smaller the item $\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)$) and the higher the local reachability densities of the k-nearest neighbors of 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 Figure 12.8, 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 o, the meaning of $LOF(o)$ is easy to understand. Consider the objects in Figure 12.9, for example. For object o, let

$$direct_{min}(o) = \min\{reachdist_k(o' \leftarrow o) | o' \in N_k(o)\} \quad (12.15)$$

be the minimum reachability distance from o to its k-nearest neighbors. Similarly, we can define

$$direct_{max}(o) = \max\{reachdist_k(o' \leftarrow o) | o' \in N_k(o)\}. \quad (12.16)$$

We also consider the neighbors of o's k-nearest neighbors. Let

$$indirect_{min}(o) = \min\{reachdist_k(o'' \leftarrow o') | o' \in N_k(o) \text{ and } o'' \in N_k(o')\} \quad (12.17)$$

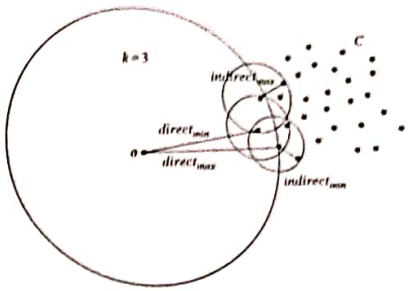


Figure 12.9 A property of $LOF(o)$.

and

$$indirect_{max}(o) = \max\{reachdist_k(o'' \leftarrow o') | o' \in N_k(o) \text{ and } o'' \in N_k(o')\}. \quad (12.18)$$

Then, it can be shown that $LOF(o)$ is bounded as

$$\frac{direct_{min}(o)}{indirect_{max}(o)} \leq LOF(o) \leq \frac{direct_{max}(o)}{indirect_{min}(o)}. \quad (12.19)$$

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

12.5 Clustering-Based Approaches

The notion of outliers is highly related to that of clusters. Clustering-based approaches detect outliers by examining the relationship between objects and clusters. Intuitively, an outlier is an object that belongs to a small and remote cluster, or does not belong to any cluster.

This leads to three general 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.

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

Example 12.15 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 Figure 12.10, 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 12.16 Clustering-based outlier detection using distance to the closest cluster. Using the *k*-means clustering method, we can partition the data points shown in Figure 12.11 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 score to the object according to the distance between the object and the center that is closest to the object. Suppose the closest center to *o* is c_o; then the distance between *o* and c_o is $dist(o, c_o)$, and the average

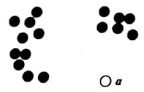


Figure 12.10 Object *a* is an outlier because it does not belong to any cluster.

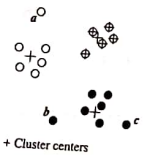


Figure 12.11 Outliers (*a*, *b*, *c*) are far from the clusters to which they are closest (with respect to the cluster centers).

distance between c_o and the objects assigned to *o* is l_{c_o}. The ratio $\frac{dist(o,c_o)}{l_{c_o}}$ measures how $dist(o, c_o)$ 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 Figure 12.11, 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 12.17.

Example 12.17 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 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., $\alpha = 90\%$) 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 12.18 Detecting outliers in small clusters. The data points in Figure 12.12 form three clusters: large clusters, C_1 and C_2, and a small cluster, C_3. Object o does not belong to any cluster.

Using CBLOF, *FindCBLOF* can identify o as well as the points in cluster C_3 as outliers. For o, the closest large cluster is C_1. The CBLOF is simply the similarity between 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 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.

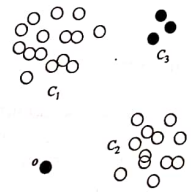


Figure 12.12 *Outliers in small clusters.*

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

12.6 Classification-Based Approaches

Outlier detection can be treated as a classification problem if a training data set with class labels is available. The general idea of classification-based outlier detection methods is to train a classification model that can distinguish normal data from outliers.

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 8 and 9). 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.

Example 12.19 Outlier detection using a one-class model. Consider the training set shown in Figure 12.13, 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 9), 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 such as in fuzzy clustering (Chapter 11). For example, *AllElectronics* 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

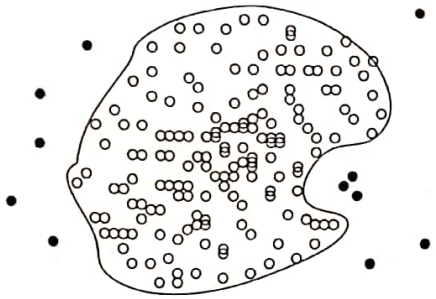
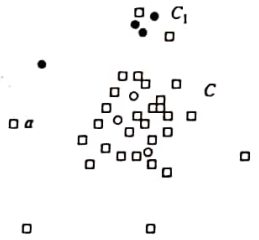


Figure 12.13 Learning a model for the normal class.



o Objects with label "normal" ● Objects with label "outlier" □ Objects without label

Figure 12.14 Detecting outliers by semi-supervised learning.

class is regarded as normal. To detect outlier cases, *AllElectronics* 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 combined to detect outliers in a semi-supervised learning way.

Example 12.20 Outlier detection by semi-supervised learning. Consider Figure 12.14, 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., a) is considered an outlier as well. ■

Example 12.22 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 *AllElectronics* 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, c, we can first locate the context of c using the attributes *age_group* and *postal_code*. We can then compare 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 c is an outlier.

Contexts may be specified at different levels of granularity. Suppose *AllElectronics* 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, c, if the corresponding context contains very few or even no other customers, the evaluation of whether 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 o belonging to cluster U_j on the contextual attributes is generated by cluster V_i on the behavior attributes. The outlier score can then be calculated as

$$S(o) = \sum_{U_j} p(o \in U_j) \sum_{V_i} p(o \in V_i) p(V_i|U_j). \tag{12.20}$$

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

12.7.2 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 the online store of *AllElectronics* 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. *AllElectronics* 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 8 and 9 on classification and the bibliographic notes for further details (Section 12.11).

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.

12.7.3 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 12.1). 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.

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 12.25 Angle-based outliers. Figure 12.15 contains a set of points forming a cluster, with the exception of c, which is an outlier. For each point o, we examine the angle $\angle xoy$ for every pair of points x, y such that $x \neq o, y \neq o$. The figure shows angle $\angle dae$ as an example.

Note that for a point in the center of a cluster (e.g., a), the angles formed as such differ widely. For a point that is at the border of a cluster (e.g., b), the angle variation is smaller. For a point that is an outlier (e.g., 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 o, we use the distance-weighted angle variance as the outlier score. That is, given a set of points, D, for a point, $o \in D$, we define the **angle-based outlier factor** (ABOF) as

$$ABOF(o) = VAR_{x,y \in D, x \neq o, y \neq o} \frac{\langle \overrightarrow{ox}, \overrightarrow{oy} \rangle}{dist(o,x)^2 dist(o,y)^2}, \tag{12.23}$$

where $\langle , \rangle$ is the scalar product operator, 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 12.11).

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.

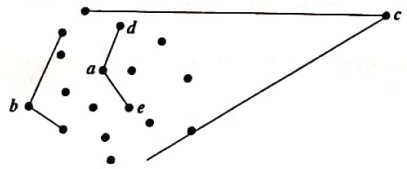


Figure 12.15 Angle-based outliers.

12.9 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*, *semi-supervised*, or *unsupervised*. Alternatively, outlier detection methods may be organized according to their assumptions regarding normal objects versus outliers. This categorization includes *statistical* methods, *proximity-based* methods, and *clustering-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 three main approaches. These include extending conventional outlier detection, finding outliers in subspaces, and modeling high-dimensional outliers.

12.10 Exercises

12.1 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?

12.2 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.

12.3 Adapt a simple semi-supervised 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.

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

12.5 Consider the nested loop approach to mining distance-based outliers (Figure 12.6). 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.

12.6 In the density-based outlier detection method of Section 12.4.3, 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.

12.7 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.

outlier detection include Jin, Tung, and Han [JTH01]; Jin, Tung, Han, and Wang [JTHW06]; and Papadimitriou, Kitagawa, Gibbons, et al. [PKG-F03]. The variations differ in how they estimate density.

The bootstrap method discussed in Example 12.17 was developed by Barbara, Li, Couto, 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, Arnold, and Prerau, 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, Wu, Jermaine, et al. [SWJR07] and Fawcet and Provost [FP97] presented a method to reduce the problem of contextual outlier detection to one of conventional outlier detection. Yi, Sidiropoulos, Johnson, Jagadish, et al. [YSJJ+00] used regression techniques to detect contextual outliers in co-evolving sequences. The idea in Example 12.22 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.

13 Data Mining Trends and Research Frontiers

As a young research field, data mining has made significant progress and covered a broad spectrum of applications since the 1980s. Today, data mining is used in a vast array of areas. Numerous commercial data mining systems and services are available. Many challenges, however, still remain. In this final chapter, we introduce the mining of complex data types as a prelude to further in-depth study readers may choose to do. In addition, we focus on trends and research frontiers in data mining. Section 13.1 presents an overview of methodologies for mining complex data types, which extend the concepts and tasks introduced in this book. Such mining includes mining time-series, sequential patterns, and biological sequences; graphs and networks; spatiotemporal data, including geospatial data, moving-object data, and cyber-physical system data; multimedia data; text data; web data; and data streams. Section 13.2 briefly introduces other approaches to data mining, including statistical methods, theoretical foundations, and visual and audio data mining.

In Section 13.3, you will learn more about data mining applications in business and in science, including the financial retail, and telecommunication industries, science and engineering, and recommender systems. The social impacts of data mining are discussed in Section 13.4, including ubiquitous and invisible data mining, and privacy-preserving data mining. Finally, in Section 13.5 we speculate on current and expected data mining trends that arise in response to new challenges in the field.

13.1 Mining Complex Data Types

In this section, we outline the major developments and research efforts in mining complex data types. Complex data types are summarized in Figure 13.1. Section 13.1.1 covers mining sequence data such as time-series, symbolic sequences, and biological sequences. Section 13.1.2 discusses mining graphs and social and information networks. Section 13.1.3 addresses mining other kinds of data, including spatial data, spatiotemporal data, moving-object data, cyber-physical system data, multimedia data, text data,