Chapter 9

Outlier Analysis: Advanced Concepts

"If everyone is thinking alike, then somebody isn't thinking."—George S. Patton

9.1 Introduction

Many scenarios for outlier analysis cannot be addressed with the use of the techniques discussed in the previous chapter. For example, the data type has a critical impact on the outlier detection algorithm. In order to use an outlier detection algorithm on categorical data, it may be necessary to change the distance function or the family of distributions used in expectation–maximization (EM) algorithms. In many cases, these changes are exactly analogous to those required in the context of the clustering problem.

Other cases are more challenging. For example, when the data is very high dimensional, it is often difficult to apply outlier analysis because of the masking behavior of the noisy and irrelevant dimensions. In such cases, a new class of methods, referred to as *subspace* methods, needs to be used. In these methods, the outlier analysis is performed in lower dimensional projections of the data. In many cases, it is hard to discover these projections, and therefore results from multiple subspaces may need to be combined for better robustness.

The combination of results from multiple models is more generally referred to as *ensemble analysis*. Ensemble analysis is also used for other data mining problems such as clustering and classification. In principle, ensemble analysis in outlier detection is analogous to that in data clustering or classification. However, in the case of outlier detection, ensemble analysis is especially challenging. This chapter will study the following three classes of challenging problems in outlier analysis:

1. *Outlier detection in categorical data:* Because outlier models use notions such as nearest neighbor computation and clustering, these models need to be adjusted to the data type at hand. This chapter will address the changes required to handle categorical data types.

2. *High-dimensional data:* This is a very challenging scenario for outlier detection because of the "curse-of-dimensionality." Many of the attributes are irrelevant and contribute to the errors in model construction. A common approach to address these issues is that of subspace outlier detection.

3. *Outlier ensembles:* In many cases, the robustness of an outlier detection algorithm can be improved with ensemble analysis. This chapter will study the fundamental principles of ensemble analysis for outlier detection.

Outlier analysis has numerous applications in a very wide variety of domains such as data cleaning, fraud detection, financial markets, intrusion detection, and law enforcement. This chapter will also study some of the more common applications of outlier analysis.

This chapter is organized as follows: Section 9.2 discusses outlier detection models for categorical data. The difficult case of high-dimensional data is discussed in Sect. 9.3. Outlier ensembles are studied in Sect. 9.4. A variety of applications of outlier detection are discussed in Sect. 9.5. Section 9.6 provides the summary.

9.2 Outlier Detection with Categorical Data

As in the case of other problems in data mining, the type of the underlying data has a significant impact on the specifics of the algorithm used for solving it. Outlier analysis is no exception. However, in the case of outlier detection, the changes required are relatively minor because, unlike clustering, many of the outlier detection algorithms (such as distance-based algorithms) use very simple definitions of outliers. These definitions can often be modified to work with categorical data with small modifications. In this section, some of the models discussed in the previous chapter will be revisited for categorical data.

9.2.1 Probabilistic Models

Probabilistic models can be modified easily to work with categorical data. A probabilistic model represents the data as a mixture of cluster components. Therefore, each component of the mixture needs to reflect a set of discrete attributes rather than numerical attributes. In other words, a generative mixture model of categorical data needs to be designed. Data points that do not fit this mixture model are reported as outliers.

The k components of the mixture model are denoted by $\mathcal{G}_1 \dots \mathcal{G}_k$. The generative process uses the following two steps to generate each point in the d-dimensional data set $\mathcal{D}$:

1. Select a mixture component with prior probability α_i, where $i \in \{1 \dots k\}$.

2. If the rth component of the mixture was selected in the first step, then generate a data point from $\mathcal{G}_r$.

The values of α_i denote the prior probabilities. An example of a model for the mixture component is one in which the jth value of the ith attribute is generated by cluster m with probability p_{ijm}. The set of all model parameters is collectively denoted by the notation Θ.

Consider a data point $\overline{X}$ containing the attribute value indices $j_1 \dots j_d$ where the rth attribute takes on the value j_r. Then, the value of the generative probability $g^{m,\Theta}(\overline{X})$ of a data point from cluster m is given by the following expression:

$$g^{m,\Theta}(\overline{X}) = \prod_{r=1}^{d} p_{rj_rm}. \tag{9.1}$$

The fit probability of the data point to the rth component is given by $\alpha_r \cdot g^{r,\Theta}(\overline{X})$. Therefore, the sum of the fits over all components is given by $\sum_{r=1}^{k} \alpha_r \cdot g^{r,\Theta}(\overline{X})$. This fit represents the likelihood of the data point being generated by the model. Therefore, it is used as the outlier score. However, in order to compute this fit value, one needs to estimate the parameters Θ. This is achieved with the EM algorithm.

The assignment (or *posterior*) probability $P(\mathcal{G}_m|\overline{X}, \Theta)$ for the mth cluster may be estimated as follows:

$$P(\mathcal{G}_m|\overline{X}, \Theta) = \frac{\alpha_m \cdot g^{m,\Theta}(\overline{X})}{\sum_{r=1}^{k} \alpha_r \cdot g^{r,\Theta}(\overline{X})}. \tag{9.2}$$

This step provides a soft assignment probability of the data point to a cluster, and it corresponds to the E-step.

The soft-assignment probability is used to estimate the probability p_{ijm}. While estimating the parameters for cluster m, the *weight* of a data point is assumed to be equal to its assignment probability $P(\mathcal{G}_m|\overline{X}, \Theta)$ to cluster m. The value α_m is estimated to be the average assignment probability to cluster m over all data points. For each cluster m, the *weighted* number w_{ijm} of records for which the ith attribute takes on the jth possible discrete value in cluster m is estimated. The value of w_{ijm} is estimated as the sum of the posterior probabilities $P(\mathcal{G}_m|\overline{X}, \Theta)$ for all records $\overline{X}$ in which the ith attribute takes on the jth value. Then, the value of the probability p_{ijm} may be estimated as follows:

$$p_{ijm} = \frac{w_{ijm}}{\sum_{\overline{X} \in \mathcal{D}} P(\mathcal{G}_m|\overline{X}, \Theta)}. \tag{9.3}$$

When the number of data points is small, the estimation of Eq. 9.3 can be difficult to perform in practice. In such cases, some of the attribute values may not appear in a cluster (or $w_{ijm} \approx 0$). This situation can lead to poor parameter estimation, or *overfitting*. The *Laplacian smoothing* method is commonly used to address such ill-conditioned probabilities. Let m_i be the number of distinct attribute values of categorical attribute i. In Laplacian smoothing, a small value β is added to the numerator, and $m_i \cdot \beta$ is added to the denominator of Eq. 9.3. Here, β is a parameter that controls the level of smoothing. This form of smoothing is also sometimes applied in the estimation of the prior probabilities α_i when the data sets are very small. This completes the description of the M-step. As in the case of numerical data, the E- and M-steps are iterated to convergence. The maximum likelihood fit value is reported as the outlier score.

9.2.2 Clustering and Distance-Based Methods

Most of the clustering- and distance-based methods can be generalized easily from numerical to categorical data. Two main modifications required are as follows:

1. Categorical data requires specialized clustering methods that are typically different from those of numerical data. This is discussed in detail in Chap. 7. Any of these models can be used to create the initial set of clusters. If a distance- or similarity-based clustering algorithm is used, then the same distance or similarity function should be used to compute the distance (similarity) of the candidate point to the cluster centroids.

2. The choice of the similarity function is important in the context of categorical data, whether a centroid-based algorithm is used or a raw, distance-based algorithm is used. The distance functions in Sect. 3.2.2 of Chap. 3 can be very useful for this task. The pruning tricks for distance-based algorithms are agnostic to the choice of distance

function and can therefore be generalized to this case. Many of the local methods, such as *LOF*, can be generalized to this case as well with the use of this modified definition of distances.

Therefore, clustering and distance-based methods can be generalized to the scenario of categorical data with relatively modest modifications.

9.2.3 Binary and Set-Valued Data

Binary data are a special kind of categorical data, which occur quite frequently in many real scenarios. The chapters on frequent pattern mining were based on these kind of data. Furthermore, both categorical and numerical data can always be converted to binary data. One common characteristic of this domain is that, while the number of attributes is large, the number of *nonzero* values of the attribute is small in a typical transaction.

Frequent pattern mining is used as a subroutine for the problem of outlier detection in these cases. The basic idea is that frequent patterns are much less likely to occur in outlier transactions. Therefore, one possible measure is to use the sum of all the supports of frequent patterns occurring in a particular transaction. The total sum is normalized by dividing with the number of frequent patterns. This provides an outlier score for the pattern. Strictly speaking, the normalization can be omitted from the final score, because it is the same across all transactions.

Let $\mathcal{D}$ be a transaction database containing the transactions denoted by $T_1 \ldots T_N$. Let $s(X, \mathcal{D})$ represent the support of itemset X in $\mathcal{D}$. Therefore, if $FPS(\mathcal{D}, s_m)$ represents the set of frequent patterns in the database $\mathcal{D}$ at minimum support level s_m, then, the frequent pattern outlier factor $FPOF(T_i)$ of a transaction $T_i \in \mathcal{D}$ at minimum support s_m is defined as follows:

$$FPOF(T_i) = \frac{\sum_{X \in FPS(\mathcal{D}, s_m), X \subseteq T_i} s(X, \mathcal{D})}{|FPS(\mathcal{D}, s_m)|}. \tag{9.4}$$

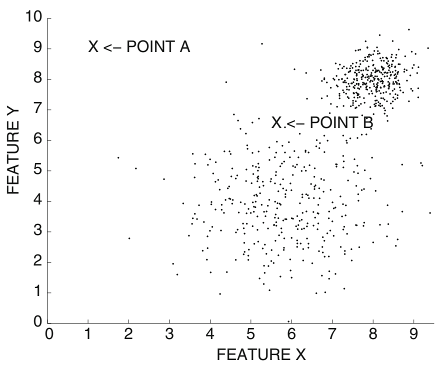
Intuitively, a transaction containing a large number of frequent patterns with high support will have a high value of $FPOF(T_i)$. Such a transaction is unlikely to be an outlier because it reflects the major patterns in the data. Therefore, lower scores indicate greater propensity to be an outlier.

Such an approach is analogous to nonmembership of data points in clusters to define outliers rather than determining the deviation or sparsity level of the transactions in a more direct way. The problem with this approach is that it may not be able to distinguish between truly isolated data points and ambient noise. This is because neither of these kinds of data points will be likely to contain many frequent patterns. As a result, such an approach may sometimes not be able to effectively determine the strongest anomalies in the data.

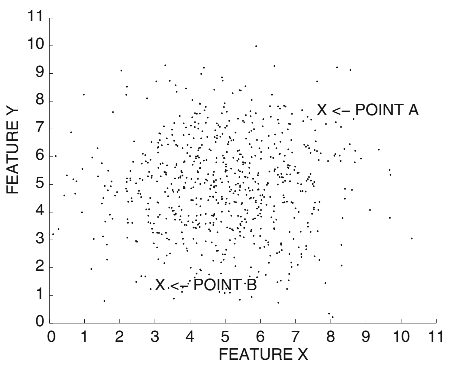
9.3 High-Dimensional Outlier Detection

High-dimensional outlier detection can be particularly challenging because of the varying importance of the different attributes with data locality. The idea is that the *causality* of an anomaly can be typically perceived in only a small subset of the dimensions. The remaining dimensions are irrelevant and only add noise to the anomaly-detection process. Furthermore, different subsets of dimensions may be relevant to different anomalies. As a result, full-dimensional analysis often does not properly expose the outliers in high-dimensional data.

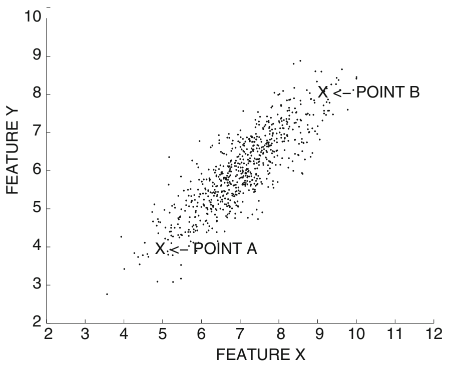
This concept is best understood with a motivating example. In Fig. 9.1, four different 2-dimensional views of a hypothetical data set have been illustrated. Each of these views



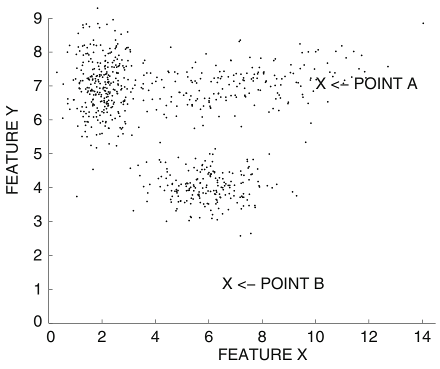
(a) View 1
Point A is outlier



(b) View 2
No outliers



(c) View 3
No outliers



(d) View 4
Point B is outlier

Figure 9.1: Impact of irrelevant attributes on outlier analysis

corresponds to a disjoint set of dimensions. As a result, these views look very different from one another. Data point A is exposed as an outlier in the first view of the data set, whereas point B is exposed as an outlier in the fourth view of the data set. However, data points A and B are not exposed as outliers in the second and third views of the data set. These views are therefore not very useful from the perspective of measuring the outlierness of *either* A or B. Furthermore, from the perspective of any *specific* data point (e.g., point A), three of the four views are irrelevant. Therefore, the outliers are lost in the random distributions within these views when the distance measurements are performed in *full* dimensionality. In many scenarios, the proportion of irrelevant views (features) may increase with dimensionality. In such cases, outliers are *lost* in low-dimensional subspaces of the data because of irrelevant attributes.

The physical interpretation of this situation is quite clear in many application-specific scenarios. For example, consider a credit card fraud application in which different features such as a customer's purchase location, frequency of purchase, and size of purchase are being tracked over time. In the case of one particular customer, the anomaly may be tracked by examining the location and purchase frequency attributes. In another anomalous customer, the size and timing of the purchase may be relevant. Therefore, all the features are useful from a *global* perspective but only a small subset of features is useful from a *local* perspective.

The major problem here is that the dilution effects of the vast number of "normally noisy" dimensions will make the detection of outliers difficult. In other words, outliers are lost in low-dimensional subspaces when full-dimensional analysis is used because of the *masking* and *dilution* effects of the noise in full-dimensional computations. A similar problem is also discussed in Chap. 7 in the context of data clustering.

In the case of data clustering, this problem is solved by defining subspace-specific clusters, or *projected* clusters. This approach also provides a natural path for outlier analysis in high dimensions. In other words, an outlier can now be defined by associating it with one or more subspaces that are *specific to that outlier*. While there is a clear analogy between the problems of subspace clustering and subspace outlier detection, the *difficulty levels* of the two problems are not even remotely similar.

Clustering is, after all, about the determination of *frequent groups* of data points, whereas outliers are about determination of *rare groups* of data points. As a rule, statistical learning methods find it much easier to determine frequent characteristics than rare characteristics of a data set. This problem is further magnified in high dimensionality. The number of possible subspaces of a d-dimensional data point is 2^d. Of these, only a small fraction will expose the outlier behavior of individual data points. In the case of clustering, dense subspaces can be easily determined by aggregate statistical analysis of the data points. This is not true of outlier detection, where the subspaces need to be explicitly explored in a way that is specific to the *individual* data points.

An effective outlier detection method would need to search the data points and dimensions in *an integrated way* to reveal the most relevant outliers. This is because different subsets of dimensions may be relevant to different outliers, as is evident from the example of Fig. 9.1. The integration of point and subspace exploration leads to a further expansion in the number of possibilities that need to be examined for outlier analysis. This chapter will explore two methods for subspace exploration, though many other methods are pointed out in the bibliographic notes. These methods are as follows:

1. *Grid-based rare subspace exploration:* In this case, rare subspaces of the data are explored after discretizing the data into a grid-like structure.

2. *Random subspace sampling:* In this case, subspaces of the data are sampled to discover the most relevant outliers.

The following subsections will discuss each of these methods in detail.

9.3.1 Grid-Based Rare Subspace Exploration

Projected outliers are determined by finding localized regions of the data in low-dimensional space that have abnormally low density. Because this is a density-based approach, a grid-based technique is the method of choice. To do so, nonempty grid-based subspaces need to be identified, whose density is very low. This is complementary to the definitions that were used for subspace clustering methods such as *CLIQUE*. In those cases, *frequent* subspaces were reported. It should be pointed out that the determination of frequent subspaces is a much simpler problem than the determination of rare subspaces, simply because there are many more rare subspaces than there are dense ones in a typical data set. This results in a combinatorial explosion of the number of possibilities, and level-wise algorithms, such as those used in *CLIQUE*, are no longer practical avenues for finding rare subspaces. The first step in all these models is to determine a proper statistical definition of rare lower dimensional projections.

9.3.1.1 Modeling Abnormal Lower Dimensional Projections

An abnormal lower dimensional projection is one in which the density of the data is exceptionally lower than the average. The concept of Z-number, introduced in the previous chapter, comes in very handy in this respect. The first step is to discretize the data. Each attribute of the data is divided into p ranges. These ranges are constructed on an *equidepth* basis. In other words, each range contains a fraction $f = 1/p$ of the records.

When k different intervals from different dimensions are selected, it creates a grid cell of dimensionality k. The expected fraction of the data records in this grid cell is equal to f^k, if the attributes are statistically independent. In practice, the data are not statistically independent, and, therefore, the distribution of the points in the cube differs significantly from the expected value. Deviations that correspond to rare regions in the data are of particular interest.

Let $\mathcal{D}$ be a database with n records, and dimensionality d. Under the independence assumption mentioned earlier, the presence or absence of any point in a k-dimensional cube is a Bernoulli random variable with probability f^k. The expected number and standard deviation of the points in a k-dimensional cube are given by $n \cdot f^k$ and $\sqrt{n \cdot f^k \cdot (1 - f^k)}$. When the value of n is large, the number of data points in a cube is a random variable that is approximated by a normal distribution, with the aforementioned mean and standard deviation.

Let $\mathcal{R}$ represent a cube in k-dimensional space, and $n_{\mathcal{R}}$ represent the number of data points inside this cube. Then, the sparsity coefficient $S(\mathcal{R})$ of the cube $\mathcal{R}$ can be computed as follows:

$$S(\mathcal{R}) = \frac{n_{\mathcal{R}} - n \cdot f^k}{\sqrt{n \cdot f^k \cdot (1 - f^k)}}. \tag{9.5}$$

A negative value of the sparsity coefficient indicates that the presence of data points in the cube is significantly lower than expected. Because $n_{\mathcal{R}}$ is assumed to fit a normal distribution, the normal distribution can be used to quantify the probabilistic level of significance of its deviation. This is only a heuristic approximation because the assumption of a normal distribution is generally not true in practice.

9.3.1.2 Grid Search for Subspace Outliers

As discussed earlier, level-wise algorithms are not practical for rare subspace discovery. Another challenge is that lower dimensional projections provide no information about the statistical behavior of combinations of dimensions, especially in the context of subspace analysis.

For example, consider a data set containing student scores in an examination. Each attribute represents a score in a particular subject. The scores in some of the subjects are likely to be highly correlated. For example, a student scoring well in a course on probability theory would likely also score well in a course on statistics. However, it would be extremely uncommon to find a student who scored well in one, but not the other. The problem here is that the *individual* dimensions provide no information about the combination of the dimensions. The rare nature of outliers makes such unpredictable scenarios common. This lack of predictability about the behavior of combinations of dimensions necessitates the use of evolutionary (genetic) algorithms to explore the search space.

Genetic algorithms mimic the process of biological evolution to solve optimization problems. Evolution is, after all, nature's great optimization experiment in which only the fittest individuals survive, thereby leading to more "optimal" organisms. Correspondingly, every solution to an optimization problem can be disguised as an individual in an evolutionary system. The measure of fitness of this "individual" is equal to the objective function value of the corresponding solution. The competing population with an individual is the group of other solutions to the optimization problem. In general, fitter organisms are more likely to survive and multiply in the population. This corresponds to the *selection* operator. The other two operators that are commonly used are *crossover* and *mutation*. Each feasible solution is encoded in the form of a string, and is considered the chromosome representation of the solution. This is also referred to as *encoding*.

Thus, each string is a solution that is associated with a particular objective function value. In genetic algorithms, this objective function value is also referred to as the *fitness function*. The idea here is that the selection operator should favor strings with better fitness (objective function value). This is similar to hill-climbing algorithms, except that genetic algorithms work with a population of solutions instead of a single one. Furthermore, instead of only checking the neighborhood (as in hill-climbing methods), genetic algorithms use crossover and mutation operators to explore a more complex concept of a neighborhood.

Therefore, evolutionary algorithms are set up to repeat the process of selection, crossover, and mutation to improve the fitness (objective) function value. As the process of evolution progresses, all the individuals in the population typically improve in fitness and also become more similar to each other. The convergence of a particular position in the string is defined as the stage at which a predefined fraction of the population has the same value for that gene. The population is said to have converged when all positions in the string representation have converged.

So, how does all this map to finding rare patterns? The relevant localized subspace patterns can be easily represented as strings of length d, where d denotes the dimensionality of the data. Each position in the string represents the index of an equi-depth range. Therefore, each position in the string can take on any value from 1 through p, where p is the granularity of the discretization. It can also take on the value $*$ ("don't care"), which indicates that the dimension is not included in the subspace corresponding to that string. The fitness of a string is based on the sparsity coefficient discussed earlier. Highly negative values of the objective function (sparsity coefficient) indicate greater fitness because one is searching for sparse subspaces. The only caveat is that the subspaces need to be nonempty to be considered fit. This is because empty subspaces are not useful for finding anomalous data points.

Consider a 4-dimensional problem in which the data have been discretized into ten ranges. Then, the string will be of length 4, and each position can take on one of 11 possible values (including the "*"). Therefore, there are a total of 11^4 strings, each of which corresponds to a subspace. For example, the string *2*6 is a 2-dimensional subspace on the second and fourth dimensions.

The evolutionary algorithm uses the dimensionality of the projection k as an input parameter. Therefore, for a d-dimensional data set, the string of length d will contain k specified position and $(d - k)$ "don't care" positions. The fitness for the corresponding solution may be computed using the sparsity coefficient discussed earlier. The evolutionary search technique starts with a population of Q random solutions and iteratively uses the processes of selection, crossover, and mutation to perform a combination of hill climbing, solution recombination, and random search over the space of possible projections. The process is continued until the population converges, based on the criterion discussed above.

At each stage of the algorithm, the m best projection solutions (most negative sparsity coefficients) are kept track of. At the end of the algorithm, these solutions are reported as the best projections in the data. The following operators are defined for selection, crossover, and mutation:

1. *Selection:* This step achieves the *hill climbing* required by the method, though quite differently from traditional hill-climbing methods. The copies of a solution are replicated by ordering them by rank and biasing them in the population in the favor of higher ranked solutions. This is referred to as *rank selection.* This results in a bias in favor of more optimal solutions.

2. *Crossover:* The crossover technique is key to the success of the algorithm because it implicitly defines the subspace exploration process. One solution is to use a uniform two-point crossover to create the recombinant children strings. It can be viewed as the process of combining the characteristics of two solutions to create two new recombinant solutions. Traditional hill-climbing methods only test an adjacent solution for a single string. The recombinant crossover approach examines a more complex neighborhood by combining the characteristics of *two* different strings, to yield two new neighborhood points.

   The two-point crossover mechanism works by determining a point in the string at random, called the crossover point, and exchanging the segments to the right of this point. This is equivalent to creating two new subspaces, by sampling subspaces from both solutions and combining them. However, such a blind recombination process may create poor solutions too often. Therefore, an optimized crossover mechanism is defined. In this mechanism, it is guaranteed that both children solutions correspond to a feasible k-dimensional projection, and the children typically have high fitness values. This is achieved by examining a subset of the different possibilities for recombination and picking the best among them.

3. *Mutation:* In this case, random positions in the string are flipped with a predefined mutation probability. Care must be taken to ensure that the dimensionality of the projection does not change after the flipping process. This is an exact analogue of traditional hill-climbing except that it is done to a population of solutions to ensure robustness. Thus, while genetic algorithms try to achieve the same goals as hill climbing, they do so in a different way to achieve better solutions.

At termination, the algorithm is followed by a postprocessing phase. In the postprocessing phase, all data points containing the abnormal projections are reported by the algorithm as the outliers. The approach also provides the relevant projections that provide the *causality* (or intensional knowledge) for the outlier behavior of a data point. Thus, this approach also has a high degree of interpretability in terms of providing the reasoning for *why* a data point should be considered an outlier.

9.3.2 Random Subspace Sampling

The determination of the rare subspaces is a very difficult task, as is evident from the unusual genetic algorithm designed discussed in the previous section. A different way of addressing the same problem is to explore many possible subspaces and examine if at least one of them contains outliers. One such well known approach is *feature bagging.* The broad approach is to repeatedly apply the following two steps:

1. Randomly, select between $(d/2)$ and d features from the underlying data set in iteration t to create a data set D_t in the tth iteration.

2. Apply the outlier detection algorithm O_t on the data set D_t to create score vectors S_t.

Virtually any algorithm O_t can be used to determine the outlier score, as long as the scores across different instantiations are comparable. From this perspective, the *LOF* algorithm is the ideal algorithm to use because of its normalized scores. At the end of the process, the outlier scores from the different algorithms need to be combined. Two distinct methods are used to combine the different subspaces:

1. *Breadth-first approach:* The ranking of the data points returned by the different algorithms is used for combination purposes. The top-ranked outliers over all the different algorithm executions are ranked first, followed by the second-ranked outliers (with repetitions removed), and so on. Minor variations could exist because of tie-breaking between the outliers within a particular rank. This is equivalent to using the best rank for each data point over all executions as the final outlier score.

2. *Cumulative sum approach:* The outlier scores over the different algorithm executions are summed up. The top-ranked outliers are reported on this basis.

At first sight, it seems that the random subspace sampling approach does not attempt to optimize the discovery of subspaces to finding rare instances at all. However, the idea here is that it is often hard to discover the rare subspaces anyway, even with the use of heuristic optimization methods. The robustness resulting from multiple subspace sampling is clearly a very desirable quality of the approach. Such methods are common in high-dimensional analysis where multiple subspaces are sampled for greater robustness. This is related to the field of *ensemble analysis*, which will be discussed in the next section.

9.4 Outlier Ensembles

Several algorithms in outlier analysis, such as the high-dimensional methods discussed in the previous section, combine the scores from different executions of outlier detection algorithms. These algorithms can be viewed as different forms of ensemble analysis. Some examples are enumerated below:

1. *Parameter tuning in LOF:* Parameter tuning in the *LOF* algorithm (cf. Sect. 8.5.2.1 of Chap. 8) can be viewed as a form of ensemble analysis. This is because the algorithm is executed over different values of the neighborhood size k, and the highest *LOF* score over each data point is selected. As a result, a more robust *ensemble* model is constructed. In fact, many parameter-tuning algorithms in outlier analysis can be viewed as ensemble methods.

2. *Random subspace sampling:* The random subspace sampling method applies the approach to multiple random subspaces of the data to determine the outlier scores as a combination function of the original scores. Even the evolutionary high-dimensional outlier detection algorithm can be shown to be an ensemble with a maximization combination function.

Ensemble analysis is a popular approach in many data mining problems such as clustering, classification, and outlier detection. On the other hand, ensemble analysis is not quite well studied in the context of outlier analysis. Furthermore, ensemble analysis is particularly important in the context of outlier analysis because of the rare nature of outliers, and a corresponding possibility of overfitting. A typical outlier ensemble contains a number of different components:

1. *Model components:* These are the individual methodologies or algorithms that are integrated to create an ensemble. For example, a random subspace sampling method combines many *LOF* algorithms that are each applied to different subspace projections.

2. *Normalization:* Different methods may create outlier scores on very different scales. In some cases, the scores may be in ascending order. In others, the scores may be in descending order. In such cases, normalization is important for meaningfully combining the scores, so that the scores from different components are roughly comparable.

3. *Model combination:* The combination process refers to the approach used to integrate the outlier score from different components. For example, in random subspace sampling, the cumulative outlier score over different ensemble components is reported. Other combination functions include the use of the maximum score, or the highest *rank* of the score among all ensembles. It is assumed that higher ranks imply greater propensity to be an outlier. Therefore, the highest rank is similar to the maximum outlier score, except that the rank is used instead of the raw score. The highest-rank combination function is also used by random subspace sampling.

Outlier ensemble methods can be categorized into different types, depending on the dependencies among different components and the process of selecting a specific model. The following section will study some of the common methods.

9.4.1 Categorization by Component Independence

This categorization examines whether or not the components are developed independently.

1. In *sequential ensembles*, a given algorithm or set of algorithms is applied sequentially, so that future applications of the algorithm are influenced by previous applications. This influence may be realized in terms of either modifications of the base data for analysis, or in terms of the specific choices of the algorithms. The final result is either a weighted combination of, or the final result of the last application of the outlier algorithm component. The typical scenario for sequential ensembles is that of *model* refinement, where successive iterations continue to improve a particular base model.

2. In *independent ensembles*, different algorithms, or different instantiations of the same algorithm, are *independently* applied to either the complete data or portions of the data. In other words, the various ensemble components are independent of the results of each other's executions.

In this section, both types of ensembles will be studied in detail.

9.4.1.1 Sequential Ensembles

In sequential ensembles, one or more outlier detection algorithms are applied sequentially to either all or portions of the data. The idea is that the result of a particular algorithmic execution may provide insights that may help refine future executions. Thus, depending upon the approach, either the data set or the algorithm may be changed in sequential executions. For example, consider the case where a clustering model is created for outlier detection. Because outliers interfere with the robust cluster generation, one possibility would be to apply the method to a successively refined data set after removing the obvious outliers through the insights gained in earlier iterations of the ensemble. Typically, the quality of the

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

Figure 9.2: Sequential ensemble framework

outliers in later iterations will be better. This also facilitates a more robust outlier detection model. Thus, the sequential nature of the approach is used for successive refinement. If desired, this approach can either be applied for a fixed number of times or used to converge to a more robust solution. The broad framework of a sequential ensemble approach is provided in Fig. 9.2.

It is instructive to examine the execution of the algorithm in Fig. 9.2 in some detail. In each iteration, a successively refined algorithm may be used on a refined data set, based on the results from previous executions. The function $f_j(\cdot)$ is used to create a refinement of the data, which could correspond to data subset selection, attribute-subset selection, or a generic data transformation method. The generality of the aforementioned description ensures that many natural variations of the method can be explored with the use of this ensemble. For example, while the algorithm of Fig. 9.2 assumes that many different algorithms $\mathcal{A}_1 \ldots \mathcal{A}_r$ are available, it is possible to select only one of them, and use it on successive modifications of the data. Sequential ensembles are often hard to use effectively in outlier analysis because of the lack of available groundtruth in interpreting the intermediate results. In many cases, the distribution of the outlier scores is used as a proxy for these insights.

9.4.1.2 Independent Ensembles

In independent ensembles, different instantiations of the algorithm or different portions of the data are executed *independently* for outlier analysis. Alternatively, the same algorithm may be applied, but with either a different initialization, parameter set, or even random seed in the case of a randomized algorithm. The *LOF* method, the high-dimensional evolutionary exploration method, and the random subspace sampling method discussed earlier are all examples of independent ensembles. Independent ensembles are more common in outlier analysis than sequential ensembles. In this case, the combination function requires careful normalization, especially if the different components of the ensemble are heterogeneous. A general-purpose description of independent ensemble algorithms is provided in the pseudocode description of Fig. 9.3.

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

Figure 9.3: Independent ensemble framework

The broad principle of independent ensembles is that different ways of looking at the same problem provide more robust results that are not dependent on specific artifacts of a particular algorithm or data set. Independent ensembles are used commonly for parameter tuning of outlier detection algorithms. Another application is that of exploring outlier scores over multiple subspaces, and then providing the best result.

9.4.2 Categorization by Constituent Components

A second way of categorizing ensemble analysis algorithms is on the basis of their constituent components. In general, these two ways of categorization are orthogonal to one another, and an ensemble algorithm may be any of the four combinations created by these two forms of categorization.

Consider the case of parameter tuning in *LOF* and the case of subspace sampling in the feature bagging method. In the first case, each model is an application of the *LOF* model with a different parameter choice. Therefore, each component can itself be viewed as an outlier analysis model. On the other hand, in the random subspace method, the same algorithm is applied to a different selection (projection) of the data. In principle, it is possible to create an ensemble with *both* types of components, though this is rarely done in practice. Therefore, the categorization by component independence leads to either *model-centered* ensembles, or *data-centered* ensembles.

9.4.2.1 Model-Centered Ensembles

Model-centered ensembles combine the outlier scores from different models built on the same data set. The example of parameter tuning for *LOF* can be considered a model-centered ensemble. Thus, it is an *independent* ensemble based on one form or categorization, and a *model-centered* ensemble, based on another.

One advantage of using *LOF* in an ensemble algorithm is that the scores are roughly comparable to one another. This may not be true for an arbitrary algorithm. For example, if the raw k-nearest neighbor distance were used, the parameter tuning ensemble would always favor larger values of k when using a combination function that picks the maximum value of the outlier score. This is because the scores across different components would not be comparable to one another. Therefore, it is crucial to use normalization during the combination process. This is an issue that will be discussed in some detail in Sect. 9.4.3.

9.4.2.2 Data-Centered Ensembles

In data-centered ensembles, different parts, samples, or functions of the data are explored to perform the analysis. A function of the data could include either a sample of the data (horizontal sample) or a relevant subspace (vertical sample). The random subspace sampling approach of the previous section is an example of a data-centered ensemble. More general functions of the data are also possible, though are rarely used. Each function of the data may provide different insights about a specific part of the data. This is the key to the success of the approach. It should be pointed out that a data-centered ensemble may also be considered a model-centered ensemble by incorporating a preprocessing phase that generates a specific function of the data as a part of the model.

9.4.3 Normalization and Combination

The final stage of ensemble analysis is to put together the scores derived from the different models. The major challenge in model combination arises when the scores across different models are not comparable with one another. For example, in a model-centered ensemble, if the different components of the model are heterogeneous, the scores will not be comparable to one another. A k-nearest neighbor outlier score is not comparable to an LOF score. Therefore, *normalization* is important. In this context, univariate extreme value analysis is required to convert scores to normalized values. Two methods of varying levels of complexity are possible:

1. The univariate extreme value analysis methods in Sect. 8.2.1 of Chap. 8 may be used. In this case, a Z-number may be computed for each data point. While such a model makes the normal distribution approximation, it still provides better scores than using raw values.

2. If more refined scores are desired, and some insights are available about "typical" distributions of outlier scores, then the mixture model of Sect. 6.5 in Chap. 6 may be used to generate probabilistically interpretable fit values. The bibliographic notes provide a specific example of one such method.

Another problem is that the ordering of the outlier scores may vary with the outlier detection algorithm (ensemble component) at hand. In some algorithms, high scores indicate greater outlierness, whereas the reverse is true in other algorithms. In the former case, the Z-number of the outlier score is computed, whereas in the latter case, the negative of the Z-number is computed. These two values are on the same scale and more easily comparable.

The final step is to combine the scores from the different ensemble components. The method of combination may, in general, depend upon the composition of the ensemble. For example, in sequential ensembles, the final outlier score may be the score from the last execution of the ensemble. However, in general, the scores from the different components are combined together with the use of a combination function. In the following, the convention assumed is that higher (normalized) scores are indicative of greater abnormality. Two combination functions are particularly common.

1. *Maximum function:* The score is the *maximum* of the outlier scores from the different components.

2. *Average function:* The score is the *average* of the outlier scores from the different components.

Both the *LOF* method and the random subspace sampling method use the *maximum* function, either on the outlier scores or the ranks[1] of the outlier scores, to avoid dilution of the score from irrelevant models. The *LOF* research paper [109] provides a convincing argument as to why the maximum combination function has certain advantages. Although the average combination function will do better at discovering many "easy" outliers that are discoverable in many ensemble components, the maximum function will do better at finding well-hidden outliers. While there might be relatively fewer well-hidden outliers in a given data set, they are often the most interesting ones in outlier analysis. A common misconception[2] is that the maximum function might overestimate the absolute outlier scores, or that it might declare normal points as outliers because it computes the maximum score over many ensemble components. This is not an issue because outlier scores are *relative*, and the key is to make sure that the maximum is computed over an equal number of ensemble components for each data point. Absolute scores are irrelevant because outlier scores are comparable on a relative basis only over a fixed data set and not across multiple data sets. If desired, the combination scores can be standardized to zero mean and unit variance. The random subspace ensemble method has been implemented [334] with a rudimentary (rank-based) maximization and an average-based combination function as well. The experimental results show that the relative performance of the maximum and average combination functions is data specific. Therefore, either the maximum or average scores can achieve better performance, depending on the data set, but the maximum combination function will be consistently better at discovering well-hidden outliers. This is the reason that many methods such as *LOF* have advocated the use of the maximum combination function.

9.5 Putting Outliers to Work: Applications

The applications of outlier analysis are very diverse, and they extend to a variety of domains such as fault detection, intrusion detection, financial fraud, and Web log analytics. Many of these applications are defined for complex data types, and cannot be fully solved with the methodologies introduced in this chapter. Nevertheless, it will be evident from the discussion in later chapters that analogous methodologies can be defined for complex data types. In many cases, other data types can be converted to multidimensional data for analysis.

9.5.1 Quality Control and Fault Detection

Numerous applications arise in outlier analysis in the context of quality control and fault detection. Some of these applications typically require simple univariate extreme value analysis, whereas others require more complex methods. For example, anomalies in the manufacturing process may be detected by evaluating the number of defective units produced by each machine in a day. When the number of defective units is too large, it can be indicative of an anomaly. Univariate extreme value analysis is useful in such scenarios.

[1]In the case of ranks, if the maximum function is used, then outliers occurring early in the ranking are assigned larger rank values. Therefore, the most abnormal data point is assigned a score (rank) of n out of n data points.

[2]This is a common misunderstanding of the Bonferroni principle [343].

Other applications include the detection of faults in machine engines, where the engine measurements are tracked to determine faults. The system may be continuously monitored on a variety of parameters such as rotor speed, temperature, pressure, performance, and so on. It is desired to detect a fault in the engine system as soon as it occurs. Such applications are often temporal, and the outlier detection approach needs to be adapted to temporal data types. These methods will be discussed in detail in Chaps. 14 and 15.

9.5.2 Financial Fraud and Anomalous Events

Financial fraud is one of the more common applications of outlier analysis. Such outliers may arise in the context of credit card fraud, insurance transactions, and insider trading. A credit card company maintains the data corresponding to the card transactions by the different users. Each transaction contains a set of attributes corresponding to the user identifier, amount spent, geographical location, and so on. It is desirable to determine fraudulent transactions from the data. Typically, the fraudulent transactions often show up as unusual combinations of attributes. For example, high frequency transactions in a particular location may be more indicative of fraud. In such cases, subspace analysis can be very useful because the number of attributes tracked is very large, and only a particular subset of attributes may be relevant to a specific user. A similar argument applies to the case of related applications such as insurance fraud.

More complex temporal scenarios can be captured with the use of time-series data streams. An example is the case of financial markets, where the stock tickers correspond to the movements of different stocks. A sudden movement, or an anomalous crash, may be detected with the use of temporal outlier detection methods. Alternatively, time-series data may be transformed to multidimensional data with the use of the data portability methods discussed in Chap. 2. A particular example is wavelet transformation. The multidimensional outlier detection techniques discussed in this chapter can be applied to the transformed data.

9.5.3 Web Log Analytics

The user behavior at different Web sites is often tracked in an automated way. The anomalies in these behaviors may be determined with the use of Web log analytics. For example, consider a user trying to break into a password-protected Web site. The sequence of actions performed by the user is unusual, compared to the actions of the majority of users that are normal. The most effective methods for outlier detection work with optimized models for sequence data (see Chap. 15). Alternatively, sequence data can be transformed to multidimensional data, using a variation of the wavelet method, as discussed in Chap. 2. Anomalies can be detected on the transformed multidimensional data.

9.5.4 Intrusion Detection Applications

Intrusions correspond to different kinds of malicious security violations over a network or a computer system. Two common scenarios are *host-based intrusions*, and *network-based intrusions*. In host-based intrusions, the operating system call logs of a computer system are analyzed to determine anomalies. Such applications are typically discrete sequence mining applications that are not very different from Web log analytics. In *network-based intrusions*, the temporal relationships between the data values are much weaker, and the data can be treated as a stream of multidimensional data records. Such applications require streaming outlier detection methods, which are addressed in Chap. 12.

9.5.5 Biological and Medical Applications

Most of the data types produced in the biological data are complex data types. Such data types are studied in later chapters. Many diagnostic tools, such as sensor data and medical imaging, produce one or more complex data types. Some examples are as follows:

1. Many diagnostic tools used commonly in emergency rooms, such as electrocardiogram (ECG), are temporal sensor data. Unusual shapes in these readings may be used to make predictions.

2. Medical imaging applications are able to store 2-dimensional and 3-dimensional spatial representations of various tissues. Examples include magnetic resonance imaging (MRI) and computerized axial tomography (CAT) scans. These representations may be utilized to determine anomalous conditions.

3. Genetic data are represented in the form of discrete sequences. Unusual mutations are indicative of specific diseases, the determination of which are useful for diagnostic and research purposes.

Most of the aforementioned applications relate to the complex data types, and are discussed in detail later in this book.

9.5.6 Earth Science Applications

Anomaly detection is useful in detecting anomalies in earth science applications such as the unusual variations of temperature and pressure in the environment. These variations can be used to detect unusual changes in the climate, or important events, such as the detection of hurricanes. Another interesting application is that of determining land cover anomalies, where interesting changes in the forest cover patterns are determined with the use of outlier analysis methods. Such applications typically require the use of spatial outlier detection methods, which are discussed in Chap. 16.

9.6 Summary

Outlier detection methods can be generalized to categorical data with the use of similar methodologies that are used for cluster analysis. Typically, it requires a change in the mixture model for probabilistic models, and a change in the distance function for distance-based models. High-dimensional outlier detection is a particularly difficult case because of the large number of irrelevant attributes that interfere with the outlier detection process. Therefore, subspace methods need to be designed. Many of the subspace exploration methods use insights from multiple views of the data to determine outliers. Most high-dimensional methods are ensemble methods. Ensemble methods can be applied beyond high-dimensional scenarios to applications such as parameter tuning. Outlier analysis has numerous applications to diverse domains, such as fault detection, financial fraud, Web log analytics, medical applications, and earth science. Many of these applications are based on complex data types, which are discussed in later chapters.

9.7 Bibliographic Notes

A mixture model algorithm for outlier detection in categorical data is proposed in [518]. This algorithm is also able to address mixed data types with the use of a joint mixture model between quantitative and categorical attributes. Any of the categorical data clustering methods discussed in Chap. 7 can be applied to outlier analysis as well. Popular clustering algorithms include k-modes [135, 278], *ROCK* [238], *CACTUS* [220], *LIMBO* [75], and *STIRR* [229]. Distance-based outlier detection methods require the redesign of the distance function. Distance functions for categorical data are discussed in [104, 182]. In particular, the work in [104] explores categorical distance functions in the context of the outlier detection problem. A detailed description of outlier detection algorithms for categorical data may be found in [5].

Subspace outlier detection explores the effectiveness issue of outlier analysis, and was first proposed in [46]. In the context of high-dimensional data, there are two distinct lines of research, one of which investigates the *efficiency* of high-dimensional outlier detection [66, 501], and the other investigates the more fundamental issue of the *effectiveness* of high-dimensional outlier detection [46]. The masking behavior of the noisy and irrelevant dimensions was discussed by Aggarwal and Yu [46]. The efficiency-based methods often design more effective indexes, which are tuned toward determining nearest neighbors, and pruning more efficiently for distance-based algorithms. The random subspace sampling method discussed in this book was proposed in [334]. An isolation-forest approach was proposed in [365]. A number of ranking methods for subspace outlier exploration have been proposed in [396, 397]. In these methods, outliers are determined in multiple subspaces of the data. Different subspaces may provide information either about different outliers or about the same outliers. Therefore, the goal is to combine the information from these different subspaces in a robust way to report the final set of outliers. The *OUTRES* algorithm proposed in [396] uses recursive subspace exploration to determine all the subspaces relevant to a particular data point. The outlier scores from these different subspaces are combined to provide a final value. A more recent method for using multiple views of the data for subspace outlier detection is proposed in [397].

Recently, the problem of outlier detection has also been studied in the context of dynamic data and data streams. The *SPOT* approach was proposed in [546], which is able to determine projected outliers from high-dimensional data streams. This approach employs a window-based time model and decaying cell summaries to capture statistics from the data stream. A set of top sparse subspaces is obtained by a variety of supervised and unsupervised learning processes. These are used to detect the projected outliers. A multiobjective genetic algorithm is employed for finding outlying subspaces from training data. The problem of high-dimensional outlier detection has also been extended to other application-specific scenarios such as astronomical data [265] and transaction data [264]. A detailed description of the high-dimensional case for outlier detection may be found in [5].

The problem of outlier ensembles is generally less well developed in the context of outlier analysis, than in the context of problems such as clustering and classification. Many outlier ensemble methods, such the *LOF* method [109], do not explicitly state the ensemble component in their algorithms. The issue of score normalization has been studied in [223], and can be used for combining ensembles. A recent position paper has formalized the concept of outlier ensembles, and defined different categories of outlier ensembles [24]. Because outlier detection problems are evaluated in a similar way to classification problems, most classification ensemble algorithms, such as different variants of bagging/subsampling, will also improve outlier detection at least from a benchmarking perspective. While the results do reflect an improved quality of outliers in many cases, they should be interpreted with caution. Many recent subspace outlier detection methods [46, 396, 397] can also be considered ensemble methods. The first algorithm on high-dimensional outlier detection [46] may also be considered an ensemble method. A detailed description of different applications of outlier analysis may be found in the last chapter of [5].

9.8 Exercises

1. Suppose that algorithm A is designed for outlier detection in numeric data, whereas algorithm B is designed for outlier detection in categorical data. Show how you can use these algorithms to perform outlier detection in a mixed-attribute data set.

2. Design an algorithm for categorical outlier detection using the Mahalanobis distance. What are the advantages of such an approach?

3. Implement a distance-based outlier detection algorithm with the use of match-based similarity.

4. Design a feature bagging approach that uses arbitrary subspaces of the data rather than axis-parallel ones. Show how arbitrary subspaces may be efficiently sampled in a data distribution-sensitive way.

5. Compare and contrast multiview clustering with subspace ensembles in outlier detection.

6. Implement any two outlier detection algorithms of your choice. Convert the scores to Z-numbers. Combine the scores using the *max* function.