Chapter 4

Proximity-Based Outlier Detection

"To lead the orchestra, you have to turn your back to the crowd." – Max Lucado

4.1 Introduction

Proximity-based techniques define a data point as an outlier when its locality (or *proximity*) is sparsely populated. The proximity of a data point may be defined in a variety of ways, which are subtly different from one another but are similar enough to merit unified treatment within a single chapter. The most common ways of defining proximity for outlier analysis are as follows:

- **Cluster-based:** The non-membership of a data point in any of the clusters, its distance from other clusters, the size of the closest cluster, or a combination of these factors are used to quantify the outlier score. The clustering problem has a complementary relationship to the outlier detection problem in which points either belong to clusters or they should be considered outliers.
- **Distance-based:** The distance of a data point to its k-nearest neighbor (or other variant) is used in order to define proximity. Data points with large k-nearest neighbor distances are defined as outliers. Distance-based algorithms typically perform the analysis at a much more detailed granularity than the other two methods. On the other hand, this greater granularity often comes at a significant computational cost.
- **Density-based:** The *number* of other points within a specified local region (grid region or distance-based region) of a data point, is used in order to define local density. These local density values may be converted into outlier scores. Other kernel-based methods or statistical methods for density estimation may also be used. The major difference between clustering and density-based methods is that clustering methods partition the data *points*, whereas density-based methods partition the data *space*.

Clearly, all these techniques are closely related because they are based on some notion of *proximity* (or similarity). The major difference is at the detailed level of *how* this proximity

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_4

is defined. These different ways of defining outliers may have different advantages and disadvantages. In many cases, the distinctions between these different classes of methods become blurred when the outlier scores are defined using[1] more than one of these concepts. This chapter addresses these issues in a unified way.

One major difference between distance-based and the other two classes of methods lies in the level of *granularity* at which the analysis is performed. In both clustering- and density-based methods, the data is pre-aggregated before outlier analysis by either partitioning the points or the space. The data points are compared to the distributions in this pre-aggregated data for analysis. On the other hand, in distance-based methods, the k-nearest neighbor distance to the *original data points* (or a similar variant) is computed as the outlier score. Thus, the analysis in nearest-neighbor methods is performed at a more detailed level of granularity than clustering methods. Correspondingly, these methods provide different trade-offs between effectiveness and efficiency for data sets of different sizes. Nearest-neighbor methods may require $O(N^2)$ time to compute all k-nearest neighbor distances for a data set with N records, unless indexing or pruning techniques are used to speed up the computations. However, indexing and pruning techniques generally work well only in some restricted settings such as lower-dimensional data sets. Furthermore, pruning is not designed for outlier score computation, and it can only be used in settings in which the binary labels (indicating whether points are outliers) need to be reported. In spite of these disadvantages, nearest-neighbor methods remain exceedingly popular. This is because such methods can often provide more detailed and accurate analysis, especially for smaller data sets in which robust clustering or density analysis is not possible. Thus, the particular choice of the model depends on the nature of the data and its size.

Proximity-based methods are naturally designed to detect both noise and anomalies, although different methods are suited to these different kinds of outliers. For example, weak definitions of proximal sparsity, such as the non-membership of data points in clusters are naturally designed to detect weak outliers (or noise), whereas large levels of deviation or sparsity in terms of density- or distance-based definitions can also detect strong outliers (or anomalies). These methods are extremely popular because of their intuitive simplicity and interpretability. In fact, a number of methods for intuitive exploration and explanation of outliers [318] are based on proximity-centered definitions. Because of the simplicity of the underlying methods, they can be easily generalized to almost all types of data such as time-series data, sequence data, or graph data.

This chapter is organized as follows. Section 4.2 discusses methods for using clusters in outlier analysis. Section 4.3 discusses distance-based methods for outlier detection. Density-based methods are discussed in section 4.4. The limitations of proximity-based outlier detection are discussed in section 4.5. Section 4.6 presents the conclusions and summary.

4.2 Clusters and Outliers: The Complementary Relationship

A well-known complementary relationship exists between clustering and outlier detection. A simplistic view would be that every data point is either a member of a cluster or an outlier. In clustering, the goal is to partition the points into dense subsets, whereas in outlier detection, the goal is to identify points that do not seem to fit naturally in these

[1]It will be discussed later in this chapter, that the well-known LOF method [96] can be interpreted either as a distance-based or density-based method, depending on how it is presented.

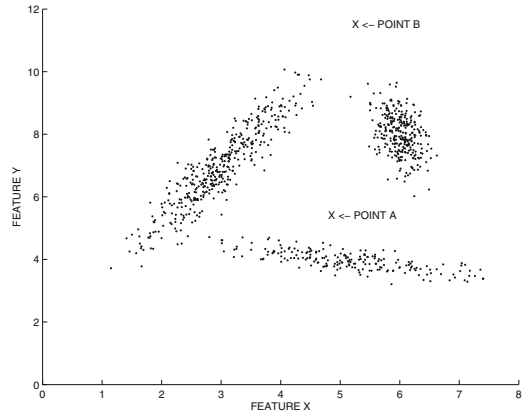


Figure 4.1: The example of Figure 2.9 revisited: Proper distance computations can detect better outliers

dense subsets. In fact, most clustering algorithms report outliers as a side-product of their analysis.

However, it is important to understand that using only the complementary relationship of points to clusters in order to define outliers results in the discovery of *weak* outliers or noise. This is because non-membership of data points in clusters is a rather blunt hammer to measure the *level of* deviation of a data point from the normal patterns. For example, a data point that is located at the fringes of a large cluster is very different from one that is completely isolated from all the other clusters. Furthermore, *all* data points in very small clusters may sometimes also be considered outliers. Therefore, when using clustering for outlier detection, a more nuanced approach (than that of cluster non-membership) is used for computing the outlier scores.

A simple definition of the outlier score may be constructed by using the distances of data points to cluster centroids. Specifically, the distance of a data point to its closest cluster centroid may be used as a proxy for the outlier score of a data point. Since clusters may be of different shapes and orientations, an excellent distance measure to use is the *Mahalanobis distance*, which scales the distance values by local cluster variances along the directions of correlation. Consider a data set containing k clusters. Assume that the rth cluster in d-dimensional space has a corresponding d-dimensional row vector $\overline{\mu_r}$ of attribute-wise means, and a $d \times d$ co-variance matrix Σ_r. The (i, j)th entry of this matrix is the local covariance between the dimensions i and j in that cluster. Then, the squared Mahalanobis distance $\mathcal{MB}(\overline{X}, \overline{\mu_r}, \Sigma_r)^2$ between a data point $\overline{X}$ (expressed as row vector) and the cluster distribution with centroid $\overline{\mu_r}$ and covariance matrix Σ_r is defined as follows:

$$\mathcal{MB}(\overline{X}, \overline{\mu_r}, \Sigma_r)^2 = (\overline{X} - \overline{\mu_r})\Sigma_r^{-1}(\overline{X} - \overline{\mu_r})^T \tag{4.1}$$

After the data points have been scored with the local Mahalanobis distance, any form of extreme-value analysis can be applied to these scores to convert them to binary labels.

One can also view the Mahalanobis distance as an adjusted Euclidean distance between a point and the cluster centroid after some transformation and scaling. Specifically, the point and the centroid are transformed into an axis system defined by the principal component directions (of the cluster points). Subsequently, the squared distance between the candidate outlier point and cluster centroid is computed along each of the new axes defined by these

principal components, and then divided by the variance of the cluster points along that component. The sum of these scaled values over all the components provides the squared Mahalanobis distance. The effect of the Mahalanobis distance is to provide statistical normalization based on the characteristics of a particular data locality. Even small distances along directions in which cluster variances are small may be *statistically* significant within that data locality. Similarly, large distances along directions in which cluster variances are large may not be statistically significant within that data locality. Such an approach will yield more refined results than a global use of the Euclidean distance because it is better tailored to the data locality at hand. This is evident from the example illustrated in Figure 4.1, in which the data point 'A' is more obviously an outlier than data point 'B' because the latter could be (weakly) related to one of the elongated clusters. However, this subtle distinction cannot be detected with the use of the Euclidean distance, according to which the data point 'A' is closest to the nearest cluster centroid. It is noteworthy that the use of the Mahalanobis distance achieves similar goals of local normalization as achieved by some other local density-based methods discussed later in this chapter (like LOF and LOCI).

The outlier scoring criterion should always be tied closely to the objective function that is optimized in the clustering algorithm. When the Mahalanobis distance is used for scoring, it should also be used within the clustering process for distance computations. For example, the Mahalanobis k-means algorithm [33] can be used in the clustering phase. Therefore, in each assignment iteration, data points are assigned to clusters based on their Mahalanobis distance to the various cluster centroids. As a result, the clustering process will be sensitive to the different shapes and orientations of the underlying clusters (as in Figure 4.1). In fact, the EM algorithm discussed in Chapter 2 can be considered a soft version of a Mahalanobis k-means algorithm [23]. Note that the term in the exponent of the Gaussian distribution for each mixture component of the probabilistic model in Chapter 2 is the (squared) Mahalanobis distance. Furthermore, the fit value computed by the EM algorithm is generally dominated by the exponentiated Mahalanobis distance to the nearest cluster centroid. The Mahalanobis k-means algorithm converts the soft probabilities into hard assignments. Thus, cluster-based outlier analysis methods are hard avatars of the (soft) probabilistic mixture models introduced in Chapter 2.

In addition to distance-based criteria, it is common to use cluster cardinality as a component of the outlier score. For example, the negative logarithm of the fraction of points in the nearest cluster can be used as a component of the outlier score. One can create two separate N-dimensional vectors of scores based on the distance and cardinality criteria, standardize each of the vectors to unit variance, and then add them. The cardinality criterion is especially popular in histogram-based methods in which the space is partitioned into regions of roughly equal size. It is noteworthy that histogram-based methods are variants of clustering methods. Incorporation of cluster cardinality into the scores is helpful in distinguishing small groups of clustered outliers from normal points occurring in larger clusters. The identification of clustered anomalies can be achieved even more effectively by using a minimum threshold on the number of data points in each cluster. An example is illustrated in Figure 4.2, where the use of a threshold of 4 is sufficient to identify the three isolated data points. Such outliers are common in real applications, because the same (rare) process might generate these outliers multiple times, albeit a small number of times. In general, clustering methods are much better than histogram-based methods in handling clustered anomalies because they partition the data *points* rather than the data *space* in a more flexible way; they can therefore detect and adjust for these types of small clusters more easily during the partitioning process.

Clustering-based methods naturally have a high variability of prediction depending on

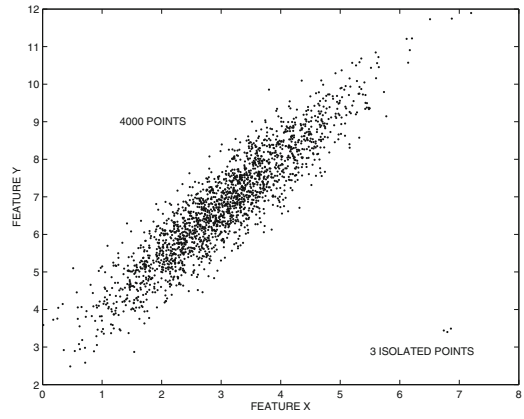


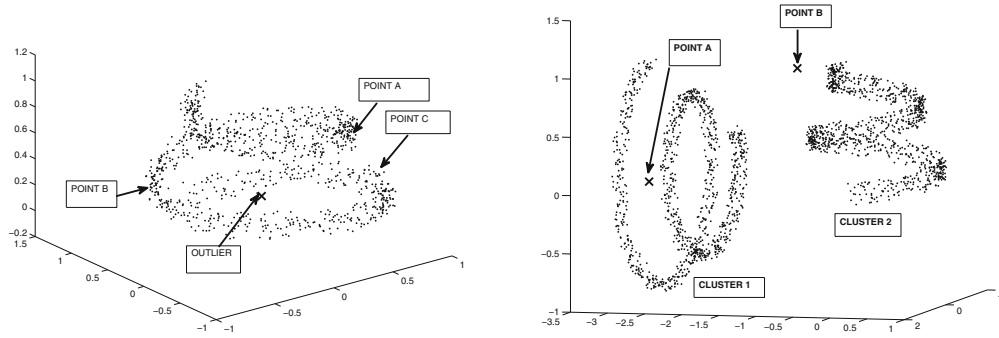
Figure 4.2: The example of Figure 1.5 revisited: Proper combination of global and local analysis in proximity-based methods can identify such outliers

the specific choice of model, randomized initialization, or parameter setting. As discussed in Chapter 6, this type of variability is a theoretical indication of a suboptimal detector *in expectation*. To improve the performance, it is often advisable to use an average of the outlier scores from multiple clusterings (with different parameter settings) to obtain better results. Even in cases where the optimal parameter setting is known, it is helpful to average the (normalized) outlier scores over different runs of a randomized clustering algorithm to obtain good results. A deterministic clustering algorithm can be suitably randomized by running it on samples of the data, using different initialization points, or explicitly randomizing specific steps of the algorithm. In general, sufficient randomization is often more important than the quality of the base clustering method. A particularly useful method in this context is the use of *extremely-randomized clustering forests* [401]. Even though the scores from a single application of clustering are often suboptimal, the use of this type of ensemble approach provides surprisingly good results. More details on such methods are provided in Chapter 6.

One interesting property of clustering ensembles is that the type of outlier discovered will be sensitive to the type of clustering that is used. For example, a subspace clustering will yield a subspace outlier (cf. section 5.2.4 of Chapter 5); a correlation-sensitive clustering will yield correlation sensitive outliers, and a locally-sensitive clustering will yield locally-sensitive outliers. By combining different base clustering methods, diverse types of outliers may be discovered. One can even extend this broad approach to other data types. Clustering ensembles have been used for discovering edge outliers [17] in graph data (see section 12.3.2.3 of Chapter 12).

4.2.1 Extensions to Arbitrarily Shaped Clusters

The discussion in the previous section on the use of the (local) Mahalanobis distance shows that the computation of distances to the nearest cluster centroid should be sensitive to the shape of the corresponding cluster. Although the Mahalanobis computation is effective for the case of elliptically shaped (Gaussian) clusters, it is not quite as effective for the case of clusters that are of arbitrary and non-convex shape. Examples of such clusters are illustrated in Figure 4.3. In the case of Figure 4.3(a) (cf. Figure 3.6(a) of Chapter 3), the entire data is arranged into a single, spiral-like, global *manifold*. The case of Figure 4.3(b)



(a) Global data distribution (b) Local data distributions

Figure 4.3: Different clusters might have different shapes. The sparsification of similarity matrices is crucial for creating embeddings that respect the varying cluster shapes in different localities.

is even more challenging because different localities of the data contain clusters of different shapes. It is also noteworthy that some of the outliers in Figures 4.3(a) and (b) are actually placed *inside* sparse regions of the non-convex clusters. An example of such an outlier is point 'A' of cluster 1 in Figure 4.3(b). The point 'A' is much closer to the centroid of cluster 1 than many of the other points belonging to the cluster. Clearly, we need some type of data transformation that can map the points to a new space in which such outliers are exposed.

The case of Figure 4.3(a) is somewhat simpler than 4.3(b), because the entire data set is arranged in a single global distribution in the former but not in the latter. For the case of Figure 4.3(a) it suffices to use methods like nonlinear principal component (or kernel principal component) analysis to map the points to a new space in which Euclidean distances can be used effectively. Such an approach is discussed in section 3.3.8 of Chapter 3. However, such methods require some modifications to adjust for the effect of varying data locality in Figure 4.3(b). Before reading further, the reader is advised to revisit the nonlinear PCA approach discussed in section 3.3.8 of Chapter 3.

As in the case of nonlinear PCA, we can use the largest eigenvectors of an $N \times N$ kernel similarity matrix $S = [s_{ij}]$ to embed the data into a multidimensional space in which Euclidean distance functions can be used effectively for clustering. The (i, j)th entry of S is equal to the kernel similarity between the ith and jth data points. In this sense, the kernel similarity functions described in section 3.3.8.1 of Chapter 3 provide a good starting point. The Gaussian kernel is typically preferred and the bandwidth is set to the median distance between sampled pairs of data points. However, these methods are best suited to discovering *global* embeddings of the data for cases like Figure 4.3(a) in which the entire data set is arranged in a single distribution. There is, therefore, a crucial modification in the construction of these similarity matrices in order to handle the varying distributions of the data in different localities (as in Figure 4.3(b)). These modifications, which are derived from the notion of *spectral clustering* [378], are those of similarity matrix *sparsification* and *local normalization* [297, 378]:

- **Sparsification:** In this case, we retain the computed similarities in S for the entry (i, j), if i is among the k-nearest neighbors of j, or if j is among the k-nearest neighbors of i. Otherwise such entries of S are set to 0. This step helps in reducing the similarity

between points from different clusters, albeit in a noisy way. Therefore, it helps in creating embeddings in which different clusters dominate different dimensions of the embedding. After performing this step, the matrix S becomes sparse, because most values of s_{ij} are 0s.

- **Local normalization:** This step is useful for performing a type of local normalization that helps[2] in adjusting to the varying density of different local regions. This step is, however, optional. Let ρ_i be the sum of the similarities in the ith row of S. Note that ρ_i represents a kind of local density near the ith data point because it is defined by the sum of its similarities to its neighbors. Each similarity value s_{ij} is then divided by the geometric mean of ρ_i and ρ_j [297]. In other words, we set $s_{ij} \Leftarrow s_{ij}/\sqrt{\rho_i \cdot \rho_j}$.

Subsequently, the top-m eigenvectors of the matrix S are extracted and are stacked in the columns of an $N \times m$ matrix D'. Typically, the value of m is quite small such as 10 to 15, although it might be data set dependent. Each column of the matrix D' is scaled to unit norm. This matrix provides the m-dimensional representation of the N data points. The value of m should be roughly set to the number of clusters that are extracted. Subsequently, the data in D' is clustered into m different clusters using the k-means algorithm on this new representation. All points are assigned to their closest cluster even if they seem like outliers. By clustering the data in the transformed representation, clusters of arbitrary shape can be discovered.

The distance of each point to its closest centroid is reported as the outlier score. However, while computing the outlier scores, it is important to use a larger number of eigenvectors than m because outliers are often emphasized along the small eigenvectors. Clustering can be performed using only a small number of eigenvectors but anomalies are often (but not always) hidden along the smaller eigenvectors. Therefore, all non-zero eigenvectors of S are extracted (while throwing away the nonzero eigenvalues caused by obvious numerical errors). This results in an $N \times n$ representation D_n, where $n \gg m$. The rows of matrix D_n are partitioned into the points belonging to the various clusters, which were discovered using the m-dimensional representation. Let the data matrices containing the n-dimensional representations of these clusters be denoted by $D_n^{(1)} \ldots D_n^{(c)}$. The columns of each $D_n^{(j)}$ are scaled[3] to unit norm in order to compute the *local* Mahalanobis distance in the *embedded* space. The n-dimensional representations of the aforementioned cluster centroids are constructed by computing the means of the n-dimensional points in each cluster $D_n^{(j)}$ (even though the clustering itself was performed in m-dimensional space). The squared Euclidean distance of each point to its cluster centroid in this n-dimensional space is reported as the outlier score.

One problem in the use of such methods is that some of the entries of the similarity matrix S are noisy. For example, an entry with high similarity between points from different clusters is noisy. This is possible in the original representation because it is hard to compute similarities accurately with kernel similarity functions that are dependent on the Euclidean distances in the original space (like the Gaussian kernel). Even a small number of such noisy entries can seriously hurt the spectral embedding. An approach is discussed in [475] for iteratively clustering the points and using the clusters to correct the noisy entries in S. This process is repeated to convergence. However, the algorithm in [475] uses a k-nearest neighbor method for scoring the points rather than the cluster-centroid distance.

[2]Although we do not describe the rationale for this local normalization in detail, it is similar to that described in section 4.4 for the Local Outlier Factor (LOF) algorithm.

[3]Some of the columns may again need to be removed, as they might have zero (local) variance. Therefore, some clusters might have less than n dimensions.

Robustness can be obtained by using cluster ensemble methods in which the number of clusters, number of eigenvectors, sparsity level and kernel bandwidth are varied within a modest range over different executions of the basic method. The final outlier score of a point is obtained by using the average score over different ensemble components. One can also construct the clusters over samples of data points to improve diversity and the efficiency of the clustering process.

4.2.1.1 Application to Arbitrary Data Types

An important advantage of such spectral methods is that they can be applied to arbitrary data types as long as a similarity function can be defined between the objects. One can use any of the available kernel similarity methods in the literature that have been defined for different data types such as time series, strings, and graphs [33]. Once the similarity function has been defined, one can create a *sparsified* similarity matrix by removing the edges with low weight. The eigenvectors of this matrix are used to create an embedding and perform the clustering. For each data point, its nearest distance to the closest cluster centroid is used to define its outlier score.

4.2.2 Advantages and Disadvantages of Clustering Methods

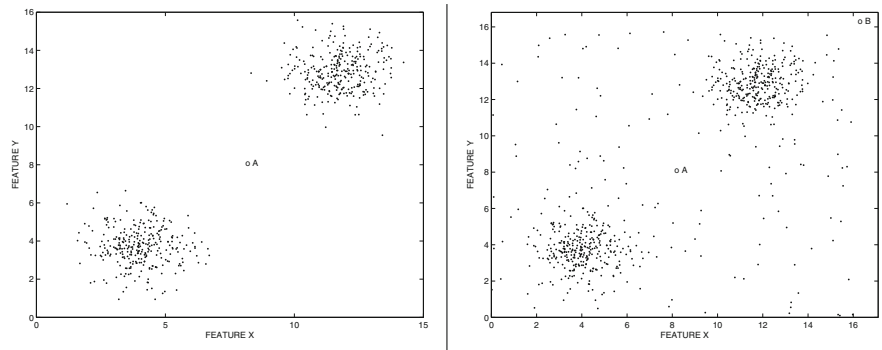
An important advantage of clustering methods is that they are relatively fast compared to the (more popular) distance-based methods. Distance-based methods have a running time that is quadratic in data dimensionality. On the other hand, many fast clustering algorithms exist in various data domains. One can also use the spectral methods discussed earlier in this section to discover outliers embedded near arbitrarily shaped clusters, or for arbitrary data types by defining appropriate similarity functions.

The main disadvantage of clustering methods is that they might not always provide insights at the required level of detail in smaller data sets. The granularity of outlier analysis methods is generally better when using distance computations directly with respect to the original data points, rather than with respect to aggregated representatives such as cluster centroids. Therefore, clustering methods are most effective when the number of available data points is sufficiently large; in such cases, these methods also have efficiency advantages. Another issue with clustering methods is that the scores have a high level of variability between different randomized executions or parameter choices. Therefore, averaging the (standardized) vector of outlier scores from different executions is essential to obtain robust results.

4.3 Distance-Based Outlier Analysis

Distance-based methods are a popular class of outlier-detection algorithms across a wide variety of data domains, and define outlier scores on the basis of *nearest neighbor distances*. The simplest example is the case in which the k-nearest neighbor distance of a point is reported as its outlier score. Because of the simplicity of this definition, it is often easy to generalize this technique to other data types. While this chapter focuses on multidimensional numerical data, such methods have been generalized to almost all other domains such as categorical data, text data, time-series data, and sequence data. The later chapters of this book will present distance-based methods for those cases.

Distance-based methods work with the natural assumption that the k-nearest neighbor distances of outlier data points are much larger than those of normal data points. A major



(a) No noise (b) With noise

Figure 4.4: The example of Figure 1.1 re-visited: Nearest neighbor algorithms may be more effective than clustering-based algorithms in noisy scenarios because of better granularity of analysis

difference between clustering and distance-based methods is in the *granularity* of the analytical process. Distance-based methods generally have a higher granularity of analysis as compared to clustering-based methods. This property of distance-based methods can enable a more refined ability to distinguish between weak and strong outliers in noisy data sets. For example, in the case of Figure 4.4, a clustering-based algorithm will not be able to distinguish between noise and anomalies easily. This is because the distance to the nearest cluster centroid for the data point 'A' will remain the same in Figures 4.4(a) and (b). On the other hand, a k-nearest neighbor algorithm will distinguish between these situations because the noisy data points will be included among the distance evaluations. On the other hand, the clustering approach will not be able to distinguish between these situations quite as well because the cluster centroids are relatively insensitive to the noise in the underlying data. Of course, it is also possible to modify cluster-based methods to include the effects of noise. In those cases, the two approaches converge to very similar schemes. This is because the two types of methods are closely related.

Distance-based methods are also able to identify isolated clusters of closely related outliers. For example, in order to identify a small (anomalous) cluster containing k_0 data points, one needs to use a value of $k \geq k_0$ in the k-nearest neighbor algorithm. Although such anomalies can also be identified by clustering methods by setting a threshold on the number of points in each cluster, such points may sometimes enter clusters and bias the corresponding cluster centroids. This can affect the outlier scoring process in a detrimental way.

The most general output of distance-based methods is in the form of scores. However, if the outlier score of each data point is required, then the (vanilla version of the) algorithm requires operations *exactly* proportional to N^2. In the binary decision version of identifying *whether* a data point is an outlier, it is possible to use various types of pruning and indexing structures to substantially speed up the approach. In the following, we will discuss algorithms for both types of outputs.

4.3.1 Scoring Outputs for Distance-Based Methods

The distance-based outlier score of a point is based on its kth nearest-neighbor distance to the remaining data set. There are two simple variations of this scoring mechanism cor-

responding to the exact k-nearest neighbor and the average k-nearest neighbor detectors. Most of the earliest methods for outlier detection focused on the use of the exact k-nearest neighbor detector.

Definition 4.3.1 (Exact k-Nearest Neighbor Score) *In a database $\mathcal{D} = \overline{X_1} \ldots \overline{X_N}$, the outlier score of any data point $\overline{X_i}$ is equal to its kth nearest neighbor distance to points in $\mathcal{D} - \{\overline{X_i}\}$.*

Note that the scored point $\overline{X_i}$ is itself not included among the k-nearest neighbors in order to avoid overfitting. For example, if we used $k = 1$ and allowed the inclusion of the candidate point among the 1-nearest neighbors, every point would be its own nearest neighbor and the outlier scores of all points would be 0. The exclusion of the candidate point from among the neighbors avoids this situation.

The main problem with this definition is that it is difficult to know the "correct" value of k for any particular data point. In unsupervised problems like outlier detection, there is often no way of parameter tuning with methods like cross-validation, because such methods require knowledge of the ground-truth. An alternative [58] that is more robust to varying choices of k is the *average k-nearest neighbor detector*, which is also referred to as the *weighted k-nearest neighbor detector*.

Definition 4.3.2 (Average k-Nearest Neighbor Score) *Consider a database $\mathcal{D} = \overline{X_1} \ldots \overline{X_N}$. The outlier score of any data point $\overline{X_i}$ is equal to its average distance to its k nearest neighbors in $\mathcal{D} - \{\overline{X_i}\}$.*

In general, if we know the "correct" value of k, the exact k-nearest neighbor tends to give better results than that given by the best value of k for the average k-nearest neighbor detector. However, in unsupervised problems like outlier detection, it is impossible to know the correct value of k for any particular algorithm, and an analyst might use a range of values of k. For example an analyst might test the algorithm with equally spaced values of k in $[1, N/10]$. In such cases, the average k-nearest neighbor method is less sensitive to different choices of k, because it effectively averages the exact k-nearest neighbor scores over a range of different values of k. A related approach, which is rarely employed, is the use of the *harmonic* average instead of the arithmetic average.

Definition 4.3.3 (Harmonic k-Nearest Neighbor Score) *Consider a database $\mathcal{D} = \overline{X_1} \ldots \overline{X_N}$. The outlier score of any data point $\overline{X_i}$ is equal to the harmonic mean of its distances to its k nearest neighbors in $\mathcal{D} - \{\overline{X_i}\}$.*

Care must be taken to remove repeated points from the data set in order to use this approach robustly, because the harmonic mean of any set of numbers containing 0 is always 0. Harmonic averages are always dominated by smaller distances (as compared to arithmetic averages), and therefore using a large value of the parameter k is advisable for greater robustness. In fact, one can set $k = N$ in the case of harmonic averaging and still obtain high-quality scores. For example, if we set $k = N$ in an (arithmetically) averaged k-nearest neighbor detector, then only multivariate extreme values will be discovered, and isolated central points might be ignored. On the other hand, the harmonically averaged scores at $k = N$ will also be able to discover isolated central points in the data, especially for large data sets. The reason for this behavior of the harmonic average is rooted in its connection to density-based methods (cf. section 4.4.4.1). The main advantage with the harmonic average is that we now have a *parameter-free* detector by setting $k = N$, and the results are still of high quality. Although harmonic nearest neighbor detectors remain unexplored in the literature, their potential is significant.

Computing the scores of all the data points is generally computationally intensive. All pairs of distances between data points need to be computed. Therefore, exactly $O(N^2)$ operations are required. This can be very expensive when the number of data points is large. For example, even for data sets containing a few hundred thousand points, the approach may not return results in a reasonable amount of time. Although it is often claimed that one can use indexing structures for efficiently finding k-nearest neighbors in $O(N \cdot \log(N))$ time, the approach is useful for only low-dimensional data sets (i.e., dimensionality less than 10). This is because index structures are not very effective at pruning in the high-dimensional case. Furthermore, such index structures have large constant overheads, which further hurts performance.

A more effective approach is to pre-select a sample of the data points. All N data points are scored with respect to this sample after excluding the candidate point from the sample (if needed). The results can be averaged over various samples in order to improve the quality of the results with the ensemble [35]. This is especially the case for the less stable variants such as harmonic averaging.

4.3.2 Binary Outputs for Distance-Based Methods

Although score-based outputs are more general than binary outputs, they have limited practical application beyond the binary problem of discovering which points are outliers. The advantage of using binary outputs is that it is possible to prune many of the $O(N^2)$ computations. Therefore, only the top-scored points are reported as outliers, and we do not care about the scores of non-outlier points. This can be achieved either by specifying a minimum threshold on the nearest-neighbor distance [317] (score) or by using a maximum threshold on the *rank* of the k-nearest neighbor distance [456]. The former parametrization presents a challenge to the analyst in selecting[4] a (possibly nonintuitive) value of an absolute distance threshold up front. The original threshold-based definition of distance-based outliers [317] was based on parameterizing it with fraction f and distance-threshold β:

Definition 4.3.4 (Score Threshold-Based Distance Outliers) *An object O in a data set $\mathcal{D}$ is a $DB(f, \beta)$ outlier, if at least fraction f of the objects in $\mathcal{D}$ lies greater than distance β from O.*

Note that score-based algorithms have a single parameter k corresponding to the kth nearest neighbor, whereas binary-thresholding algorithms have two parameters f and β. The parameter f is virtually equivalent to using a parameter like k in the original definition. Instead of using a fraction f, we can use the exact kth nearest neighbor distance by setting $k = \lceil N(1 - f) \rceil$. For uniformity in discussion throughout this chapter, we restate this definition in terms of the kth nearest-neighbor distance:

Definition 4.3.5 (Score Threshold-Based Distance Outliers) *An object in a data set $\mathcal{D}$ is an outlier, if its exact kth-nearest neighbor distance is at least β.*

A second definition [456] is based on top-r thresholding rather than the thresholding of the absolute values of the scores. Therefore, the points are *ranked* in decreasing order of the k-nearest neighbor distance. The top-r such data points are reported as outliers. Therefore, the threshold is on the distance *rank* rather than the distance *value*.

[4]It is possible to compute the outlier scores of a sample of the data points and set the estimate based on the mean and standard deviation of these scores.

Definition 4.3.6 (Rank Threshold-Based Distance Outliers) *An object in a data set $\mathcal{D}$ is an outlier, if its exact kth-nearest neighbor distance is among the top-r such values in the data set.*

The two definitions are virtually identical, except for the parameter choice presented to the user. In fact, for every choice of distance threshold β, an appropriate value of r can be selected in order to yield the same result in the two cases.

Note that the most straightforward approach for solving this problem is to first compute all pairwise k-nearest neighbor distances with a nested loop approach. Subsequently, the appropriate thresholding criterion can be applied to report the relevant points as outliers. However, this naive approach is computationally inefficient. After all, the main advantage of computing binary outputs over scoring outputs is that we can couple the outlier detection process with a pruning methodology to make the approach more efficient.

In all the aforementioned definitions, we have used the exact k-nearest neighbor distance because of the preponderance of this definition in the literature. However, all the aforementioned definitions and some of the associated pruning methods can be generalized to the average k-nearest neighbor distance. In the following, we will discuss various pruning methods for the exact k-nearest neighbor distance, and also investigate their generalizations to the average k-nearest neighbor distance.

4.3.2.1 Cell-Based Pruning

The *cell-based* technique [317] is based on the score-wise thresholding of Definition 4.3.5. The method is designed for the exact k-nearest neighbor distance, and it cannot be easily generalized to the average k-nearest neighbor distance. In the cell-based technique, the data space is divided into cells, the width of which is a function of the distance threshold β and the data dimensionality d. Specifically, each dimension is divided into cells of width at most $\frac{\beta}{(2\cdot\sqrt{d})}$. This odd value of the width is chosen to force certain distance-centric properties of the points in the cells, which are exploited for pruning and efficient processing. The approach is best explained in the 2-dimensional case. Consider the 2-dimensional case, in which successive grid-points are at a distance of at most $\beta/(2\cdot\sqrt{2})$. An important point to be kept in mind is that the number of grid-cells is based on a partitioning of the data *space*, and is independent of the number of data points. This is an important factor in the efficiency of the approach for low dimensional data, in which the number of grid-cells is likely to be modest. On the other hand, this approach is not suited to data of higher dimensionality.

For a given cell, its L_1 neighbors are the cells reached by crossing a single cell-to-cell boundary. Note that two cells touching at a corner are also L_1 neighbors. The L_2 neighbors are the cells obtained by crossing either 2 or 3 boundaries. A particular cell marked X, along with its set of L_1 and L_2 neighbors, is illustrated in Figure 4.5. It is evident that an interior cell has 8 L_1 neighbors and 40 L_2-neighbors. Then, the following properties can be immediately observed.

1. The distance between a pair of points in a cell is at most $\beta/2$.
2. The distance between a point and a point in its L_1 neighbor is at most β.
3. The distance between a point and a point in its L_r neighbor (where $r > 2$) is at least β.

The only cells for which immediate conclusions cannot be drawn, are those in L_2. This represents the region of uncertainty for the data points in a particular cell. All the distance computations in the approach are performed between pairs of points in this region

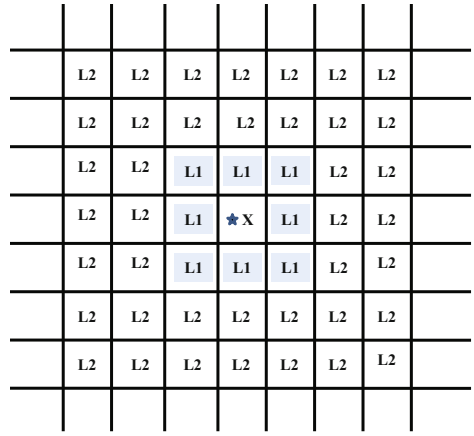


Figure 4.5: Cell-based partitions of data space

of uncertainty. However, further pruning gains are possible with a number of rules that are able to efficiently identify some of the points as outliers or non-outliers without having to materialize all these distance computations. These are as follows:

1. If more than k data points are contained in a cell *together with* its L_1 neighbors, then *none* of these data points are outliers.

2. If no more than k data points are contained in a cell 'A' and its L_1 and L_2 neighbors, then *all* points in cell 'A' are outliers.

The approach uses these various properties and rules to label points as outliers or non-outliers in an efficient way. The first step is to directly label all points in cells containing more than k points as non-outliers because of the first rule. Furthermore, all neighbor cells of such cells exclusively contain non-outliers. In order to obtain the full pruning power of the first rule, the sum of the points in each cell and its L_1 neighbors is computed. If the total number is greater than k, then all these points are labeled as non-outliers as well.

Next, the pruning power of the second rule is leveraged. For each cell 'A' containing at least one data point, the sum of the number of points in it, and the numbers in its L_1 and L_2 neighbors is computed. If this number is no more than k, then all points in cell 'A' are labeled as outliers. At this point, many cells may been labeled as outliers or non-outliers. This provides major pruning gains.

The data points in cells that have not been labeled as either outlier or non-outlier need to have their k-nearest neighbor distance computed explicitly. Even for such data points, the computation of the k-nearest neighbor distances can be made faster with the use of the cell structure. Consider a cell 'A' which has not been labeled as a pure outlier or pure non-outlier cell so far. Such cells may possibly contain a mixture of outliers and non-outliers. The main region of uncertainty for the data points in cell 'A' are the set of points in the L_2-neighbors of this cell 'A'. It cannot be known whether the points in the L_2 neighbors of 'A' are within the threshold distance of β for the points in cell 'A'. Explicit distance computations are required in order to determine the number of points within the threshold β for the data points in cell 'A'. Those data points for which no more than k points in L_1 and L_2 have distance less than β are declared outliers. Note that distance computations need to be explicitly performed only from points in cell 'A' to the points in the L_2 neighbors

of cell 'A.' This is because all points in L_1 neighbors are already known to be at a distance less than β from any point in 'A', and all points in L_r for $r > 2$ are already known to be at least a distance of β from any point in 'A'. Therefore, an additional level of savings is achieved in the distance computations.

The aforementioned description is for the 2-dimensional case. The approach can also be extended to higher dimensions. The main difference for the d-dimensional case is in terms of the width of a cell (which is now $\beta/(2 \cdot \sqrt{d})$) and the definition of L_2-neighbors. In the case of 2-dimensional data, the L_2-neighbors of a cell are defined as is non-neighbor cells that are at most three cells away. In the general case of higher dimensions, L_2 is defined as the set of cells that are at most $\lceil 2 \cdot \sqrt{d} \rceil$ cells away but not immediate neighbors. All other steps of the algorithm remain identical. However, for the high-dimensional case, this approach becomes increasingly expensive because the number of cells increases exponentially with data dimensionality. Thus, this approach is generally suited to low-dimensional data.

In many cases, the data sets may not be available in main memory, but may be stored on disk. The data-access efficiency therefore becomes a concern. It has been shown in [317] how this approach can be applied to disk-resident data sets with the use of clustered page reads. This algorithm requires at most three passes over the data. More details are available in [317].

4.3.2.2 Sampling-Based Pruning

Sampling methods are extremely flexible, in that they can handle the score-based thresholding (Definition 4.3.5) or the rank-based thresholding (Definition 4.3.6). Furthermore, they can be used with either the exact k-nearest neighbor detector or the average k-nearest neighbor detector. They are extremely efficient at pruning, and can also be used as robust ensemble methods [32] (cf. Chapter 6). For these reasons, sampling methods should be considered the first line of attack for efficient distance-based outlier detection. In the following discussion, we will use the rank-based thresholding definition together with an exact k-nearest neighbor detector. However, the generalization of the methodology to any other combination of thresholding and detection is straightforward and therefore omitted.

The first step is to select a sample $\mathcal{S}$ of size $s \ll N$ from the data $\mathcal{D}$, and compute all pairwise distances between the data points in sample $\mathcal{S}$ and those in database $\mathcal{D}$. There are a total of $N \cdot s$ such pairs. This process requires $O(N \cdot s) \ll O(N^2)$ distance computations. Thus, for each of the sampled points in $\mathcal{S}$, the k-nearest neighbor distance is already known exactly. The top rth ranked outlier in sample $\mathcal{S}$ is determined, where r is the number of outliers to be returned. The score of the rth rank outlier provides a *lower* bound[5] L on the rth ranked outlier score over the entire data set $\mathcal{D}$. For the data points in $\mathcal{D} - \mathcal{S}$, only an *upper bound* $V^k(\overline{X})$ on the k-nearest neighbor distance is known. This upper bound is equal to the k-nearest neighbor distance of each point in $\mathcal{D} - \mathcal{S}$ to the sample $\mathcal{S} \subset \mathcal{D}$. However, if this upper bound $V^k(\overline{X})$ is no larger than the lower bound L already determined, then such a data point $\overline{X} \in \mathcal{D} - \mathcal{S}$ can be excluded from further consideration as a top-r outlier. Typically, this will result in the removal of a large number of outlier candidates from $\mathcal{D} - \mathcal{S}$ immediately, as long as the underlying data set is clustered well. This is because most of the data points in clusters will be removed, as long as at least one point from each cluster is included in the sample $\mathcal{S}$, and at least r points in $\mathcal{S}$ are located in somewhat sparse regions. This can often be achieved with modest values of the sample size s in real-world data sets. After removing these data points from $\mathcal{D} - \mathcal{S}$, the remaining set of points is $\mathcal{R} \subseteq \mathcal{D} - \mathcal{S}$. The k-nearest neighbor approach can be applied to a much smaller set of candidates $\mathcal{R}$.

[5]Note that higher k-nearest neighbor distances indicate greater outlierness.

The top-r ranked outliers in $\mathcal{R} \cup \mathcal{S}$ are returned as the final output. Depending on the level of pruning already achieved, this can result in a very significant reduction in computational time, especially when $|\mathcal{R} \cup \mathcal{S}| \ll |\mathcal{D}|$.

Early Termination Trick with Nested Loops

The approach discussed in the previous section can be improved even further by speeding up the second phase of computing the k-nearest neighbor distances of each data point in $\mathcal{R}$. The idea is that the computation of the k-nearest neighbor distance of any data point $\overline{X} \in \mathcal{R}$ need not be followed through to termination once it has been determined that $\overline{X}$ cannot possibly be among the top-r outliers. In such cases, the scan of the database $\mathcal{D}$ for computation of the k-nearest neighbor of $\overline{X}$ can be terminated early.

Note that one already has an estimate (upper bound) $V^k(\overline{X})$ of the k-nearest neighbor distance of every $\overline{X} \in \mathcal{R}$, based on distances to sample $\mathcal{S}$. Furthermore, the k-nearest neighbor distance of the rth best outlier in $\mathcal{S}$ provides a lower bound on the "cut-off" required to make it to the top-r outliers. This lower-bound is denoted by L. This estimate $V^k(\overline{X})$ of the k-nearest neighbor distance of $\overline{X}$ is further tightened (reduced) as the database $\mathcal{D} - \mathcal{S}$ is scanned, and the distance of $\overline{X}$ is computed to each point in $\mathcal{D} - \mathcal{S}$. Because this running estimate $V^k(\overline{X})$ is always an upper bound on the true k-nearest neighbor distance of $\overline{X}$, the process of determining the k-nearest neighbor of $\overline{X}$ can be terminated as soon as $V^k(\overline{X})$ falls below the known lower bound L on the top-r outlier distance. This is referred to as *early termination* and provides significant computational savings. Then, the next data point in $\mathcal{R}$ can be processed. In cases where early termination is not achieved, the data point $\overline{X}$ will almost[6] always be among the top-r (current) outliers. Therefore, in this case, the lower bound L can be tightened (increased) as well, to the new rth best outlier score. This will result in even better pruning when the next data point from $\mathcal{R}$ is processed to determine its k-nearest neighbor distance value. To maximize the benefits of pruning, the data points in $\mathcal{R}$ should not be processed in arbitrary order. Rather, they should be processed in decreasing order of the initially sampled estimate $V^k(\cdot)$ of the k-nearest neighbor distances (based on $\mathcal{S}$). This ensures that the outliers in $\mathcal{R}$ are found early on, and the bound L is tightened as fast as possible for even better pruning. Furthermore, in the inner loop, the data points $\overline{Y}$ in $\mathcal{D} - \mathcal{S}$ can be ordered in the opposite direction, based on *increasing* value of $V^k(\overline{Y})$. Doing so ensures that the k-nearest neighbor distances are updated as fast as possible, and the advantage of early termination is maximized. The nested loop approach can also be implemented without the first phase[7] of sampling, but such an approach will not have the advantage of proper ordering of the data points processed. Starting with an initial lower bound L on the rth best outlier score obtained from the sampling phase, the nested loop is executed as follows:

[6] We say "almost," because the very last distance computation for $\overline{X}$ may bring $V(\overline{X})$ below L. This scenario is unusual, but might occasionally occur.

[7] Most descriptions in the literature omit the first phase of sampling, which is very important for efficiency maximization. A number of implementations in time-series analysis [311] do order the data points more carefully but not with sampling.

for each $\overline{X} \in \mathcal{R}$ **do begin**
  for each $\overline{Y} \in \mathcal{D} - \mathcal{S}$ **do begin**
    Update current k-nearest neighbor distance estimate $V^k(\overline{X})$ by
      computing distance of $\overline{Y}$ to $\overline{X}$;
    if $V^k(\overline{X}) \leq L$ **then** terminate inner loop;
  endfor
  if $V^k(\overline{X}) > L$ **then**
    include $\overline{X}$ in current r best outliers and update L to
      the new rth best outlier score;
endfor

Note that the k-nearest neighbors of a data point $\overline{X}$ do not include the data point itself. Therefore, care must be taken in the nested loop structure to ignore the trivial cases where $\overline{X} = \overline{Y}$ while updating k-nearest neighbor distances.

4.3.2.3 Index-Based Pruning

Indexing and clustering are two other common forms of data localization and access. Therefore, it is natural to explore, whether some of the traditional clustering methods or index structures can be used in order to improve the complexity of distance-based computations. The original approach [456] uses the ranking-based thresholding (Definition 4.3.6) with an exact k-nearest neighbor detector.

As in the case of sampling-based pruning, an upper bound $V^k(\overline{X})$ of the k-nearest neighbor distance of candidate data point $\overline{X} = (x_1 \ldots x_d)$ is progressively tightened in conjunction with pruning. The approach uses minimum bounding rectangles of sets of points to estimate distance bounds of the candidate $\overline{X}$ to any point in the set. These bounds can be used to prune these sets without explicit distance computation in a manner that is discussed later. Let R be a minimum bounding rectangle, where the lower and upper bounds along the ith dimension are denoted by $[r_i, r_i']$. Then, the minimum distance min_i of x_i along the ith dimension to any point in the minimum bounding rectangle R is potentially 0, if $x_i \in [r_i, r_i']$. Otherwise, the minimum distance is $\min\{|x_i - r_i|, |x_i - r_i'|\}$. Therefore, by computing this minimum value along each dimension, it is possible to estimate the total minimum bound to the entire rectangle R by $\sum_{i=1}^{d} min_i^2$. Similarly, the maximum distance max_i of $\overline{X}$ along the ith dimension to the bounding rectangle R is given by $\max\{|x_i - r_i|, |x_i - r_i'|\}$. The corresponding total maximum value can be estimated as $\sum_{i=1}^{d} max_i^2$. The aforementioned bounds can be used in conjunction with index structures such as the R^*-Tree [78] for estimating the k-nearest neighbor distance of data points. This is because such index structures use minimum bounding rectangles in order to represent the data at the nodes. In order to determine the outliers in the data set, the points are processed one by one in order to determine their k-nearest neighbor distances. The highest r such distances are maintained dynamically over the course of the algorithm. A branch-and-bound pruning technique is used on the index structure in order to determine the value of $V^k(\overline{X})$ efficiently. When the minimum distance estimate to a bounding rectangle is larger than the value of $V^k(\overline{X})$, then the bounding rectangle obviously does not contain any points which would be useful for updating the value of $V^k(\overline{X})$. Such subtrees of the R^*-Tree can be completely pruned from consideration.

Aside from the index-based pruning, individual data points can also be discarded from consideration early. The score (i.e., k-nearest neighbor distance) of the rth best outlier found so far is denoted by $Dmin$. Note that $Dmin$ is exactly the same as the bound L used in the

previous section on sampling. The estimate of $V^k(\overline{X})$ for a data point $\overline{X}$ is monotonically decreasing with algorithm progression, as better nearest neighbors are found. When this estimate falls below $Dmin$, the point $\overline{X}$ can be discarded from consideration.

Many variations of this basic pruning technique have been proposed for different data domains [311]. Typically, such algorithms work with a nested loop structure, in which the outlier scores of candidate data points are computed one by one *in a heuristically ordered outer loop* which approximates a decreasing level of outlier score. For each point, the nearest neighbors are computed in the inner loop in a *heuristic ordering* that approximates increasing distance to the candidate point. The inner loop can be abandoned, when its currently approximated nearest neighbor distance is less than the rth best outlier found so far ($V^k(\overline{X}) < Dmin$).

A good heuristic ordering in the outer and inner loops can ensure that the data point can be discarded from consideration early. Different tricks can be used to determine this heuristic ordering in various application settings. In many cases, the method for finding the heuristic ordering in the outer loop uses the complementarity of the clustering and outlier detection problem, and orders the data points on the basis of the cardinality of the clusters they are contained in. Data points in clusters containing very few (or one) point(s) are examined first. Typically, a very simple and efficient clustering process is used to create the outer-loop ordering. The method for finding the heuristic ordering in the inner loop typically requires a fast approximation of the k-nearest neighbor ordering, and is dependent upon the specific data domain or application. An adaptation of this approach for time-series data [311] is discussed in Chapter 9.

Partition-based Speedup

The approach discussed above may require the reasonably accurate computation of $V^k(\overline{X})$ for a large number of points, if the bound estimation process discussed above is not sufficiently robust. This can still be expensive in spite of pruning. In practice, the value of r is quite small, and many data points $\overline{X}$ can be excluded from consideration without estimating $V^k(\overline{X})$ explicitly. This is achieved by using clustering [456] in order to perform partitioning of the data space, and then analyzing the data at this level of granularity. A partition-based approach is used to prune away those data points which could not possibly be outliers in a computationally efficient way. This is because the partitioning represents a less granular representation of the data, which can be processed at lower computational costs. For each partition, a lower bound and an upper bound on the k-nearest neighbor distances of *all* included data points is computed. If the aforementioned upper bound on the k-nearest neighbor distance estimate of any point in the partition is less than a current value of $Dmin$, then the *entire partition* can be pruned from consideration of containing *any* outlier points. The partition-based method also provides a more efficient way for approximating $Dmin$. First, the partitions are sorted by decreasing lower bound. The first l partitions containing at least r points are determined. The lower bound on the l-th partition provides an approximation for $Dmin$. The upper and lower bound for each partition is computed using the minimum bounding rectangle of the node in the index structure containing the points. More savings may be obtained by using the fact that the distances from each (unpruned) candidate data point $\overline{X}$ do not need to be computed to data points in partitions that are guaranteed to be further away than the current upper bound on the k-nearest neighbor distance of the point $\overline{X}$ (or its containing partition).

Thus, this analysis is performed at a less detailed level of granularity. This makes its efficiency closer to that of clustering-based methods. In fact, the partitions are themselves

generated with the use of a clustering algorithm such as BIRCH [611]. Thus, this approach prunes many data points, and then works with a much smaller set of candidate partitions on which the analysis is performed. This greatly improves the efficiency of the approach. The exact details of computing the bounds on the partitions use the aforementioned estimations on the minimum and maximum distances to the bounding rectangles of different partitions and are discussed in detail in [456]. Because of the close relationship between distance-based and clustering methods, it is natural to use clustering methods to improve the approximations on the k-nearest neighbor distance. A number of other techniques in the literature use clustering in order to achieve better pruning and speedups in distance-based algorithms [58, 219, 533].

4.3.3 Data-Dependent Similarity Measures

The k-nearest neighbor method can be paired with other types of similarity and distance functions. Unlike the Euclidean distance, data-dependent similarity measures depend not just on the pair of points at hand but also on the statistical distribution of the other points [33]. This can help in incorporating various data-dependent characteristics into the similarity function such as *locality-sensitivity*, *correlation-sensitivity*, or both. An intuitive example of locality-sensitivity is that the same pair of Caucasians would be considered more similar in Asia than in Europe [548].

A well-known locality-sensitive similarity measure is the *shared nearest neighbor similarity measure* [287], in which the intersection cardinality of the k-nearest neighbor sets of two points is used as the similarity value. The basic idea here is that in a dense region, two points have to be very close[8] in order to have a large number of common nearest neighbors, whereas in a sparse region it is possible for reasonably distant points to have a larger number of common neighbors. The main drawback of the shared nearest-neighbor measure is that we now have two parameters for the number of nearest neighbors, which can potentially be different; one parameter is for the similarity computation and the other is for the distance-based outlier detection algorithm. A larger number of parameters creates uncertainty in unsupervised problems because of difficulty in setting the parameters optimally. The computation of the shared nearest-neighbor measure is also expensive. It is possible to compute this similarity in a robust and efficiently way by repeatedly sampling the data, computing the measure, and averaging the measure over different samples. When computing the similarity with sampled data, the shared nearest neighbor distances are computed between all pairs of points (whether they are sampled or not), but only the number of shared neighbors within the sample are counted.

The pairwise Mahalanobis distance, which adapts the Mahalanobis method to compute distances between pairs of points, is a correlation-sensitive distance measure. This measure is equivalent to computing the Euclidean distance between pairs of points after transforming the data to uncorrelated directions with PCA and normalizing each dimension in the transformed data to unit variance. One advantage of this measure is that it uses the correlation structure of the underlying data. Two data points at unit Euclidean distance in the (untransformed) data will have larger Mahalanobis distance if they were aligned along a low-variance direction rather than a high-variance direction in the untransformed data. One can also use the *nonlinear* Mahalanobis method in conjunction with a kernel or spectral similarity matrix to make the distance function sensitive to arbitrary shapes in the

[8] As discussed in section 4.4.1, this intuition is directly used by locality-sensitive algorithms like the Local Outlier Factor (LOF) method. Therefore, by appropriately changing the similarity function in the exact k-nearest neighbor algorithm, one can achieve similar goals as locality-sensitive algorithms like LOF.

underlying data distribution [475].

Random forests can be used to compute data-dependent similarity [99, 491, 555] because of their ability [359] to define data locality in a distribution-dependent way. In the unsupervised setting, the basic idea is to generate synthetic data for the outlier class, and treat the provided data set as the normal class. As discussed in [491], the outliers are generated uniformly at random between the minimum and maximum range of each attribute. A random forest is constructed on the synthetically labeled data. The similarity between a pair of instances (say, A and B) can be defined as either (i) the number of trees in the random forest in which they occur in the same leaf node, or (ii) the average length of the common path traversed by the two instances A and B in each tree. The main advantage of this approach is that it is very efficient to compute the similarity between a pair of instances once the random forest has been constructed.

In principle, any hierarchical representation of the data that preserves its clustering structure can be used to measure similarity with the aforementioned approach as long as the clustering adjusts well to varying local density and correlations. Other unsupervised methods for creating random trees include the use of randomized hierarchical clustering methods [401, 555] and the isolation forest [548]. The former [401] also provides the option to create a "bag-of-words" representation of the data containing the identifiers of the leaf and/or internal tree nodes to which data points are assigned. The computation of hamming distance on this representation is almost equivalent to random-forest similarity measures. Data-dependent similarity measures for categorical data are discussed in section 8.4.2 of Chapter 8.

4.3.4 ODIN: A Reverse Nearest Neighbor Approach

Most of the distance-based methods directly use the k-nearest neighbor distribution in order to define outliers. A different approach is to use the *number of reverse k-nearest neighbors* in order to define outliers [248]. Therefore, the concept of a reverse k-nearest neighbor is defined as follows:

Definition 4.3.7 *A data point p is a reverse k-nearest neighbor of q, if and only if q is among the k-nearest neighbors of p.*

Data points that have large k-nearest neighbor distances, will also have few reverse neighbors, because they will lie among the k-nearest neighbors of very few data points. Thus, an outlier is defined as a point for which the number of reverse k-nearest neighbors is less than a pre-defined user-threshold.

The reverse nearest neighbor approach can also be easily understood in terms of the underlying k-nearest neighbor graph. Consider a graph in which the nodes correspond to the data points. A directed edge (p, q) is added to the graph if and only if q is among the k-nearest neighbors of p. Thus, every node has an outdegree of k in this graph. However, the in-degree of the nodes may vary, and is equal to the number of reverse k-nearest neighbors. The nodes with few reverse k-nearest neighbors are declared outliers. Alternatively, the number of reverse k-nearest neighbors may be reported as the outlier *score*. Smaller values are more indicative of a greater degree of outlierness. The reverse nearest-neighbor method is also referred to as *Outlier Detection using In-degree Number (ODIN)*. Like the shared nearest neighbor similarity in the previous section, all methods that use the k-nearest neighbor graph [100, 248] are locality sensitive.

This approach requires the determination of all the k-nearest neighbors of each node. Furthermore, distance-based pruning is no longer possible since the nearest neighbors of

Table 4.1: Example of Outliers in NHL Player Statistics [318]

Player Name	Short-Handed Goals	Power-Play Goals	Game-Winning Goals	Game-Tying Goals	Games Played
Mario Lemieux	31	8	8	0	70
Jaromir Jagr	20	1	12	1	82
John Leclair	19	0	10	2	82
R. Brind'Amor	4	4	5	4	82

each node need to be determined explicitly. Thus, the approach may potentially require $O(N^2)$ time for construction of the k-nearest neighbor graph. The other problem with the approach is that many data points might be ties in terms of the number of reverse nearest neighbors (outlier score). Both these problems can be addressed with ensemble methods [32]. Data points are repeatedly scored with a subsample of s points, where s is chosen to be a random value in a constant range. The average score over various samples is reported as the outlier score.

4.3.5 Intensional Knowledge of Distance-Based Outliers

An important issue in outlier analysis is to retain a high level of interpretability for providing intuitive explanations and insights. This is very important in many application-driven scenarios. The concept of *intensional knowledge* of distance-based outliers was first proposed in [318]. The idea is to explain the outlier behavior of objects in terms of subsets of attributes. Thus, in this case, a *minimal* bounding box on the subsets of attributes is presented in order the explain the outlier behavior of the data points. For example, consider the case of National Hockey League (NHL) player statistics, which was first presented in [318]. An example set of statistics is illustrated in Table 4.1. The sample output from [318], which explains these outliers is as follows:

Mario Lemieux	An outlier in the 1-d space of power play goals An outlier in the 2-d space of short-handed goals and game-winning goals
R. Brind'Amor	An outlier in the 1-d space of game-tying goals.

Several notions are defined in [318] in order to understand the importance of an outlier:

1. Is a particular set of attributes the minimal set of attributes in which an outlier exists?
2. Is an outlier dominated by other outliers in the data?

The intensional knowledge can be directly characterized in terms of cells, because they define the bounding rectangles along different attributes. The work in [318] proposes a number of roll-up and drill-down methods in order to define the interesting combinations of attributes for intensional knowledge. The concept of *strong* and *weak* outliers is also defined. Outliers that are defined by minimal combinations of attributes are generally considered

stronger from an intensional perspective. It should be emphasized that this definition of strong and weak outliers is specific to an intensional knowledge-based approach and is different from the more general form in which this book uses these terms (as the outlier tendency of an object).

4.3.6 Discussion of Distance-Based Methods

Distance-based methods have a number of qualitative advantages over clustering-based techniques because of the more detailed granularity of the analysis. For example, distance-based algorithms can distinguish between noise and anomalies much better than cluster-based techniques. Furthermore, distance-based methods can also find isolated groups of outliers just like clustering methods. On the other hand, clustering methods have the advantage that they can provide insights about the *local* distributions of data points for defining distances. For example, in the case of Figure 4.1, the local cluster structure can be used in order to define a *locally sensitive* Mahalanobis distance, which is much more effective at identifying outliers, than a blind application of the Euclidean metric. However, it is also possible to design a distance-based algorithm based on the Mahalanobis distance, which is also referred to as *instance-specific Mahalanobis method* [33]. These correspond to the data-dependent similarity measures discussed in section 4.3.3.

Although the density-based methods explained later in this chapter do incorporate some notions of locality, they are still unable to provide the detailed level of local insights that an effective combination of a clustering- and distance-based approach can provide. In this context, some recent research has incorporated local clustering insights into distance-based methods [7, 153, 475]. Furthermore, the efficiency advantages of clustering methods should be incorporated into generalized distance-based methods in order to obtain the best results.

4.4 Density-Based Outliers

Density-based methods use the number of points in specific regions of the space to define outliers. They are very closely related to clustering and distance-based methods. In fact, some of these algorithms can be more easily considered clustering or distance-based methods, depending on how they are presented. This is because the notions of distances, clustering, and density are closely related and inter-dependent. Many of these algorithms discover *locality-sensitive* outliers. This section discusses both local and global algorithms.

The sensitivity of distance-based outliers to data locality was first noticed in [96]. While the Figure 4.1 illustrates the general effect of data locality on both data density and cluster orientation, the work in [96, 97] specifically addresses the issue of varying local density. In order to understand this specific issue, consider the specific example of a data set with varying density in Figure 4.6. The figure contains two outliers labeled 'A' and 'B.' Furthermore, the figure contains two clusters, one of which is much sparser than the other. It is evident that the outlier 'A' cannot be discovered by a distance-based algorithm unless a smaller distance-threshold is used by the algorithm. However, if a smaller distance threshold is used, then many data points in the sparser cluster may be incorrectly declared as outliers. This also means that the *ranking returned by a distance-based algorithm is incorrect when there is significant heterogeneity in the local distributions of the data.* This observation was also noted (more generally) in section 4.2, where it was shown that the outliers in Figure 4.1 are sensitive to both the local cluster density and the cluster orientation (local attribute correlation). However, much of the work in density-based clustering has generally focused

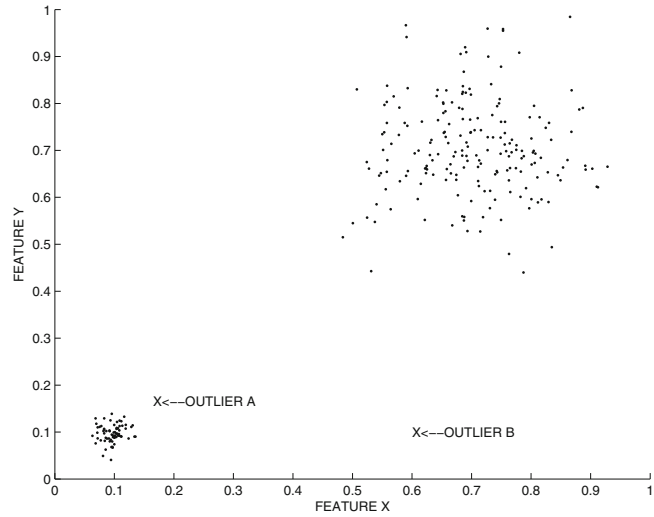


Figure 4.6: Impact of local density on outliers

on issues of varying data density rather than the varying shape and orientation of clusters.

The k-nearest neighbor algorithm can be paired with the data-dependent similarity measures of section 4.3.3 to address problems caused by local density variations. However, the LOF method [96] uses a different approach by defining the outlier score in a locally-adjusted manner.

4.4.1 LOF: Local Outlier Factor

The Local Outlier Factor (LOF) is a quantification of the outlierness of the data points, which is able to adjust for the variations in the different local densities. For a given data point $\overline{X}$, let $D^k(\overline{X})$ be its distance to the k-nearest neighbor of X, and let $L_k(\overline{X})$ be the set of points within the k-nearest neighbor distance of $\overline{X}$. Note that $L_k(\overline{X})$ will typically contain k points, but may sometimes contain more than k points because of ties in the k-nearest neighbor distance.

Then, the reachability distance $R_k(\overline{X}, \overline{Y})$ of object $\overline{X}$ with respect to $\overline{Y}$ is defined as the maximum of $dist(\overline{X}, \overline{Y})$ and the k-nearest neighbor distance of $\overline{Y}$:

$$R_k(\overline{X}, \overline{Y}) = \max\{dist(\overline{X}, \overline{Y}), D^k(\overline{Y})\}$$

The reachability distance is not symmetric between $\overline{X}$ and $\overline{Y}$. Intuitively, when $\overline{Y}$ is in a dense region and the distance between $\overline{X}$ and $\overline{Y}$ is large, the reachability distance of $\overline{X}$ with respect to it is equal to the true distance $dist(\overline{X}, \overline{Y})$. On the other hand, when the distances between $\overline{X}$ and $\overline{Y}$ are small, then the reachability distance is smoothed out by the k-nearest neighbor distance of $\overline{Y}$. The larger the value of k is, the greater the smoothing will be. Correspondingly, the reachability distances with respect to different points will also become more similar.

Then, the *average reachability distance* $AR_k(\overline{X})$ of data point $\overline{X}$ is defined as the average of its reachability distances to all objects in its neighborhood $L_k(\overline{X})$:

$$AR_k(\overline{X}) = \text{MEAN}_{\overline{Y} \in L_k(\overline{X})} R_k(\overline{X}, \overline{Y})$$

Here, the MEAN function simply represents the mean of a set of values. The work in [96] also defines the reachability density as the inverse of this value, though this particular presentation omits this step, since the LOF values can be expressed more simply and intuitively in terms of the average reachability distance $AR_k(\overline{X})$. The *Local Outlier Factor* is then simply equal to the mean ratio of $AR_k(\overline{X})$ to the corresponding values of all points in the k-neighborhood of $\overline{X}$:

$$LOF_k(\overline{X}) = \text{MEAN}_{\overline{Y} \in L_k(\overline{X})} \frac{AR_k(\overline{X})}{AR_k(\overline{Y})} \tag{4.2}$$

$$= AR_k(\overline{X}) \cdot \text{MEAN}_{\overline{Y} \in L_k(\overline{X})} \frac{1}{AR_k(\overline{Y})} \tag{4.3}$$

The use of distance ratios in the definition ensures that the local distance behavior is well accounted for in this definition. As a result, the LOF values for the objects in a cluster are often close to 1, when the data points in the cluster are homogeneously distributed. For example, in the case of Figure 4.6, the LOF values of data points in *both* clusters will be close to 1, even though the densities of the two clusters are different. On the other hand, the LOF values of *both* the outlying points will be much higher since they will be computed in terms of the ratios to the average neighbor reachability distances. The LOF scores can also be viewed as *normalized reachability distance* of a point, where the normalization factor is the *harmonic* mean of the reachability distances in its locality. For example, Equation 4.3 can be rewritten as follows:

$$LOF_k(\overline{X}) = \frac{AR_k(\overline{X})}{\text{HMEAN}_{\overline{Y} \in L_k(\overline{X})} AR_k(\overline{Y})} \tag{4.4}$$

Here, HMEAN denotes the harmonic mean of the reachability distances of all the points in its locality. In principle, we could use any type of mean in the denominator. For example, if we had used a *arithmetic* mean of the reachability distances in the denominator, the results would have a similar interpretation. One observation about the LOF method is that while it is popularly understood in the literature as a density-based approach, it can be more simply understood as a *relative* distance-based approach with smoothing. The relative distances are computed on the basis of the local distribution of reachability distances. The LOF method was originally presented in [96] as a density-based approach because of its ability to adjust to regions of varying density. The density is loosely defined as the inverse of the smoothed reachability distances of a point. This is, of course, not a precise definition of density, which is traditionally defined in terms of the number of data points within a specified area or volume. The presentation in this chapter omits this intermediate density variable, both for simplicity, and for a definition of LOF directly in terms of reachability distances. The real connection of LOF to data density lies in its insightful ability to *adjust* to varying data density with the use of *relative distances*. While this book has also classified this method as a density-based approach, it can be equivalently understood in terms of either a relaxed definition of density or distances. There are a small number of simplifications that one could make to LOF without affecting its performance very significantly:

1. Instead of using the reachability distances, one might use the raw distances.
2. Instead of using the harmonic mean in Equation 4.4, one can simply use the arithmetic mean. Another variant known as LDOF (local distance-based outlier factor) [610] uses the averaged *pairwise-distances* between points in $L_k(\overline{X})$ instead of the harmonic mean of arithmetic means.

With such modifications, the relationship to distance-based methods is more obvious. It is also possible to use data-dependent similarity measures (cf. section 4.3.3) in combination with an exact k-nearest neighbor outlier detector to achieve similar goals as LOF.

4.4.1.1 Handling Duplicate Points and Stability Issues

The use of the harmonic mean in Equation 4.4 has some interesting consequences for the stability of the algorithm because the harmonic mean is often not a very stable central representative of a set of values. The occurrence of a single value of 0 in a set of values will result in a harmonic mean of 0. This has consequences in a data set containing duplicate points (or points in dense regions that are very close to one another). For example, if *even one* of the reachability values in the denominator of Equation 4.4 is 0, the entire expression will be set to ∞. In other words, *all points in the immediate neighborhood of duplicate points are at risk of having their scores set to* ∞. It is clear that such an outcome is not desirable. Note that this problem is reduced with the use of the arithmetic mean instead of the harmonic mean in Equation 4.4. Another possibility is to use k-distinct-distances, which is essentially equivalent to removing duplicates from the data set for model construction [96]. However, removing duplicates is beneficial only if these duplicates are a result of noise or errors in the data. If the duplicates represent true densities, then the computed LOF score will be biased. Therefore, it is hard to use the solution of k-distinct-distances without a deeper semantic understanding of the duplicates in the data set. Finally, a reasonable approach is to use regularization by modifying Equation 4.4 with a small value $\alpha > 0$:

$$LOF_k(\overline{X}) = \frac{\alpha + AR_k(\overline{X})}{\alpha + \text{HMEAN}_{\overline{Y} \in L_k(\overline{X})} AR_k(\overline{Y})} \tag{4.5}$$

Intuitively, the value of α regulates the prior probability that a data point is a normal point.

Although the issue of duplicates can be effectively handled using the aforementioned methods, harmonic normalization does result in broader issues of stability with respect to the value of the parameter k. For example, it is recommended in [96] to use the maximum value of $LOF_k(\overline{X})$ over a range of different values of k as the outlier score [96]. However, at small values of k, groups of points might be close to one another by chance, which will increase the outlier scores of points in their locality. This problem can be viewed as soft version of the duplicate problem discussed earlier. Because of the unstable nature of harmonic normalization, even a *single* tightly knit group can increase the outlier scores of many points in a given locality.

This type of instability tends to make the LOF detector more sensitive to the value of k than other detectors such as the average k-nearest neighbor method. Therefore, if one can determine the best value of k for LOF, it is possible to obtain better results for LOF than those obtained using the best value of k for other detectors like the exact or average k-nearest neighbor methods. However, one must be careful in interpreting such results, because it is impossible to know the correct value of k in a particular detector for unsupervised problems like outlier detection. Furthermore, the good results of (the relatively unstable) LOF at particular values of k might simply be a manifestation of overfitting. In practice, a better performance measure is to compute various measures of the average performance over a range of parameter choices. In such cases, LOF is often outperformed by simpler detectors such as the exact k-nearest neighbor detector and the average k-nearest neighbor detector [32, 35]. Indeed, the trusty old k-nearest neighbor detectors should never be underestimated.

4.4.2 LOCI: Local Correlation Integral

An interesting method proposed in [426] uses a local density-based method for outlier analysis. The LOCI method is truly a density-based method, since it defines the density $M(\overline{X}, \epsilon)$ of a data point $\overline{X}$ in terms of the number of data points within a pre-specified radius ϵ around a point. This is referred to as the *counting neighborhood* of the data point $\overline{X}$. Correspondingly, the average density $AM(\overline{X}, \epsilon, \delta)$ in the δ-neighborhood of $\overline{X}$ is defined as the mean value of $M(\overline{X}, \epsilon)$ for all data points at a distance at most δ from $\overline{X}$. The value of δ is also referred to as the *sampling* neighborhood of $\overline{X}$, and is always larger than ϵ. Furthermore, the value of ϵ is always chosen as a constant fraction of δ, no matter what value of δ is used. The value of δ is a critical parameter in the analysis, and multiple values of this parameter are used in order to provide analytical insights at different levels of granularity. The average density in the neighborhood of $\overline{X}$ is formally defined as follows:

$$AM(\overline{X}, \epsilon, \delta) = \text{MEAN}_{[\overline{Y}:dist(\overline{X},\overline{Y})\leq\delta]} M(\overline{Y}, \epsilon) \tag{4.6}$$

Correspondingly, the *multi-granularity deviation factor* $MDEF(\overline{X}, \epsilon, \delta)$ at level δ is expressed in terms of the ratio of the densities at a point and the average density in its neighborhood:

$$MDEF(\overline{X}, \epsilon, \delta) = 1 - \frac{M(\overline{X}, \epsilon)}{AM(\overline{X}, \epsilon, \delta)} \tag{4.7}$$

Note the similarity to LOF in terms of using the local ratios while defining the outlier score of a data point. The larger the value of the MDEF is, the greater the outlier score. In order to convert the MDEF score into a binary label, the deviation $\sigma(\overline{X}, \epsilon, \delta)$ of the different values of $M(\overline{X}, \epsilon)$ within the sampling neighborhood of $\overline{X}$ is computed.

$$\sigma(\overline{X}, \epsilon, \delta) = \frac{STD_{[\overline{Y}:dist(\overline{X},\overline{Y})\leq\delta]} M(\overline{Y}, \epsilon)}{AM(\overline{X}, \epsilon, \delta)}$$

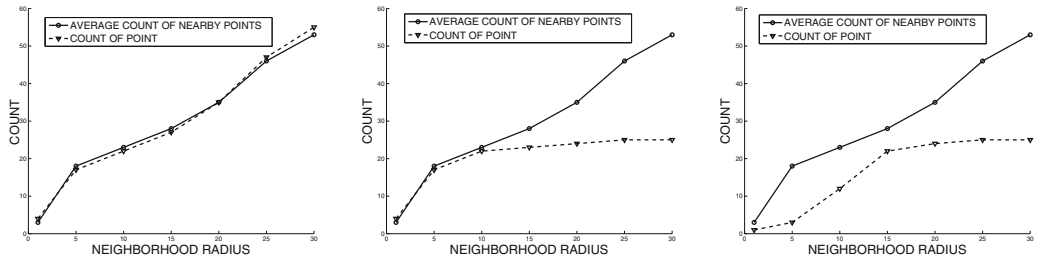
Here, the function STD computes the standard deviation over the entire sampling neighborhood. The denominator accounts for the fact that the MDEF value of Equation 4.7 is scaled by the same expression in the denominator.

The value of ϵ is always chosen to be half that of δ in order to enable fast approximate computation. Therefore, throughout this presentation, it is assumed that the value of ϵ is automatically decided by the choice of δ. Multiple values of δ are used in order to provide a multi-granularity approach for outlier analysis. These methods vary the sampling radius from a minimum radius containing at least 20 points to a maximum radius that spans most of the data. A data point is an outlier if its MDEF value is unusually large among *any* of the values computed at different granularity levels. Specifically, the value of the MDEF needs to be at least $k \cdot \sigma(\overline{X}, \epsilon, \delta)$, where k is chosen to be 3. This choice of k is common in statistical analysis with the use of the normal distribution assumption.

The algorithmic efficiency can be improved with the following modifications:

- Only a limited set of sampling neighborhoods need to be considered. In particular, if the sampling or counting neighborhoods do not change for small changes in δ, then those neighborhoods do not need to be considered.

- Fast methods for approximating the neighbor counts are also provided in [426]. This provides high-quality approximations to MDEF. It has been shown in [426], that a box count of a grid-based division of the data provides a fast approximation, when L_∞ distances are used. This approximation is also referred to as the aLOCI algorithm.



(a) Cluster point (b) Micro-clustered anomaly (c) Outstanding anomaly

Figure 4.7: Illustrative example of a simplified version of the LOCI plot without ranges shown. The ranges can further help in distinguishing when the true density of a point falls below the average density of its neighborhood. The figure is drawn for illustrative purposes only to show the typical types of trends one would see in various settings.

4.4.2.1 LOCI Plot

The LOCI plot compresses the information about a data point in a two dimensional representation, where the outlier behavior is visually interpretable from a multi-granular perspective. Since the value of $MDEF(\overline{X}, \epsilon, \delta)$ is constructed by examining the relative behavior of $M(\overline{X}, \epsilon)$ and $AM(\overline{X}, \epsilon, \delta)$ over different values of δ, it makes sense to visualize each of these quantities by separately plotting them against the sampling neighborhood δ. Therefore, the LOCI plot shows the value of δ on the X-axis against the following count-based quantities on the Y-axis:

- **Density behavior of point:** The value of $M(\overline{X}, \epsilon) = M(\overline{X}, \delta/2)$ is plotted on the Y-axis. This shows the actual density behavior of the data point $\overline{X}$ at different granularity levels.
- **Average density behavior of nearby points:** The values of $AM(\overline{X}, \epsilon, \delta)$ and $AM(\overline{X}, \epsilon, \delta) \pm STD_{\left[\overline{Y}:dist(\overline{X},\overline{Y})\leq\delta\right]} M(\overline{Y}, \epsilon)$ are plotted on the Y-axis. This shows the density behavior of the neighborhood of $\overline{X}$ (along with statistical ranges) for different granularity levels. The ranges help in identifying cases in which the actual density of the point falls significantly below the true density of its neighborhood.

When a data point is an outlier, its density behavior will often fall below that of the point neighborhood, and it might even fall below the lower end of the neighborhood-density range.

The LOCI plot provides a *visual* understanding of how the deviations of the data point relate to extreme values of the deviation at different granularity levels, and it *explains* why a particular data point may have a high MDEF value. The use of different granularity levels is helpful in adjusting the algorithm to the vagaries of different data sets. For example, in the case of Figure 4.2, any distance-based or the LOF method would need to select the value of k (for the k-nearest neighbor method) very carefully in order to identify these data points as outliers. However, the LOCI plot would always visually enable a correct point-specific level of granularity of analysis. In some cases, specific data points may show up as outliers only at particular granularity levels, which would also show up in the LOCI plot. This type of visual insight is helpful for building intuition and interpretability in unsupervised problems like outlier detection.

An illustrative example of the LOCI plot is shown in Figure 4.7. Here, we have shown a simplified version of the plot without showing the upper and lower ranges of the aver-

age neighborhood density. In other words, we have shown only the neighborhood density $AM(\overline{X}, \epsilon, \delta)$ without showing the ranges $AM(\overline{X}, \epsilon, \delta) \pm STD_{[\overline{Y}:dist(\overline{X},\overline{Y})\leq\delta]} M(\overline{Y}, \epsilon)$. These ranges are further helpful in distinguishing when the point density falls far below neighborhood density. Figure 4.7(a) shows the typical behavior of a clustered point in which the density of a point is roughly similar to the neighborhood density over all values of the radius. In the case of a clustered anomaly, which co-occurs with a small number of points, the two densities diverge only after sufficiently increasing the neighborhood size (see Figure 4.7(b)) beyond the size of the small cluster. Clustered anomalies tend to be common because of the propensity of outliers to occur in small groups. In the case of an outstanding outlier, as shown in Figure 4.7(c), the two densities diverge at the very beginning. Thus, these different plots provide insights about why specific points should be considered outliers. It is also noteworthy that the LOCI plot is specific to a single data point; therefore, it is usually leveraged only over a small set of interesting points identified by the LOCI algorithm.

4.4.3 Histogram-Based Techniques

Histograms use a space-partitioning methodology for density-based summarization. In the simplest case of univariate data, the data is discretized into bins of equal width between the minimum and maximum values, and the frequency of each bin is estimated. Data points that lie in bins with very low frequency are reported as outliers. In the context of multivariate data, the approach can be generalized in two different ways:

- The outlier scores are computed separately for each dimension, and then the scores can be aggregated.
- The discretization along each dimension can be generated at the same time, and a grid structure can be constructed. The distribution of the points in the grid structure can be used in order to create a model of the sparse regions. The data points in these sparse regions are the outliers.

In some cases, the histogram is constructed only on a sample of the points (for efficiency reasons), but all points are scored based on the frequency of a bin in which a point lies. Let $f_1 \ldots f_b$ be the (raw) frequencies of the b univariate or multivariate bins. These frequencies represent the outlier scores of the points inside these bins. Smaller values are more indicative of outlierness. In some cases, the frequency count (score) of a data point is adjusted by reducing it by 1. This is because the inclusion of the data point itself in the count can mask its outlierness during extreme value analysis. This adjustment is particularly important if a sample of the data is used to construct the histogram, but out-of-sample points are scored using the frequencies of their relevant bins in the constructed histogram. In such cases, the scores of only the in-sample points are reduced by 1. Therefore, the adjusted frequency F_j of the jth point (belonging to the ith bin with frequency f_i) is given by the following:

$$F_j = f_i - I_j \tag{4.8}$$

Here, $I_j \in \{0, 1\}$ is an indicator variable depending on whether or not the jth point is an in-sample point. Note that F_j is always nonnegative, because bins containing in-sample points always have a frequency of at least 1.

It is noteworthy the logarithm of the adjusted frequency F_j represents a log-likelihood score, which enables better extreme-value analysis for score to label conversion. For regularization purposes, we use $\log_2(F_j + \alpha)$ as the outlier score of the jth point, where $\alpha > 0$ is the

regularization parameter. To convert the scores to binary labels, a Student's t-distribution or normal distribution can be used to determine unusually low scores with extreme-value analysis. Such points are labeled as outliers. Histograms are very similar to clustering methods because they summarize the dense and sparse regions of the data in order to compute outlier scores; the main difference is that clustering methods partition the data *points*, whereas histogram methods tend to partition the data *space* into regions of equal size.

The major challenge with histogram-based techniques is that it is often hard to determine optimal histogram width well. Histograms that are too wide or too narrow will not model the frequency distribution at the level of granularity needed to optimally detect outliers. When the bins are too narrow, the normal data points falling in these bins will be identified as outliers. On the other hand, when the bins are too wide, anomalous data points may fall in high-frequency bins, and will therefore not be identified as outliers. In such a setting, it makes sense to vary the histogram width and obtain multiple scores for the same data point. These (log-likelihood) scores are then averaged over the different base detectors to obtain a final result. Like clustering, histogram-based methods tend to have a high variability in prediction depending on parameter choices, which makes them ideal candidates for ensemble methods. For example, the *RS-Hash* method (cf. section 5.2.5 of Chapter 5) varies the dimensionality and size of grid regions to score points as outliers. Similarly, some recent ensemble methods like isolation forests (cf. section 5.2.6) can be viewed as randomized histograms in which the grid regions of varying size and shapes are created in hierarchical and randomized fashion. Instead of measuring the number of points in a grid region of fixed size, an indirect measure of the expected size of the grid region required to isolate a single point is reported as the outlier score. This type of approach can avoid the problem of selecting a fixed grid size up front.

A second (related) challenge with the use of histogram-based techniques is that their space-partitioning approach makes them myopic to the presence of clustered anomalies. For example, in the case of Figure 4.2, a multivariate grid-based approach may not be able to classify an isolated group of data points as outliers, unless the resolution of the grid structure is calibrated carefully. This is because the density of the grid only depends on the data points inside it, and an isolated group of points may create an artificially dense grid cell, when the granularity of representation is high. This issue can also be partially resolved by varying the grid width and averaging the scores.

Histogram methods do not work very well in higher dimensionality because of the sparsity of the grid structure with increasing dimensionality, unless the outlier score is computed with respect to carefully chosen lower dimensional projections. For example, a d-dimensional space will contain at least 2^d grid-cells, and therefore, the number of data points expected to populate each cell reduces exponentially with increasing dimensionality. As discussed in section 4.4.5, the use of some techniques like rotated bagging [32] and subspace histograms (cf. section 5.2.5 of Chapter 5) can partially address this issue. Although histogram-based techniques have significant potential, they should be used in combination with high-dimensional subspace ensembles (see Chapters 5 and 6) for best results.

4.4.4 Kernel Density Estimation

Kernel-density estimation methods are similar to histogram techniques in terms of building density profiles. However, a smoother version of the density profile is constructed by using kernel functions rather than discrete counting. Because of the similarity between the two classes of methods, one tends to obtain similar results in the two cases, especially if ensemble methods are used to smooth out the implementation- and parameter-specific effects.

In kernel density estimation [496], which is also known as the Parzen-Rosenblatt method, a continuous estimate of the density is generated at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K'_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h that regulates the level of smoothing. The kernel density estimate, $\hat{f}(\overline{X})$, based on N data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\hat{f}(\overline{X}) = \frac{1}{N} \cdot \sum_{i=1}^{N} K'_h(\overline{X} - \overline{X_i}) \tag{4.9}$$

Thus, each discrete point $\overline{X_i}$ in the data set is replaced by a continuous function $K'_h(\cdot)$ which peaks at $\overline{X_i}$ and has a variance that is determined by the smoothing parameter h. An example of such a distribution would be the Gaussian kernel with width h for d-dimensional data:

$$K'_h(\overline{X} - \overline{X_i}) = \left(\frac{1}{h\sqrt{2\pi}}\right)^d \cdot e^{-\frac{||\overline{X} - \overline{X_i}||^2}{2h^2}} \tag{4.10}$$

The estimation error is defined by the kernel width h, which is chosen in a data-driven manner. It has been shown [496] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\hat{f}(\overline{X})$ asymptotically converges to the true density function $f(\overline{X})$, provided that the width h is chosen appropriately. For a data set containing N points, the Silverman approximation rule [496] suggests a bandwidth of $h = 1.06\,\hat{\sigma}\,N^{-1/5}$, where the $\hat{\sigma}^2$ is the estimated sample variance. This choice is, however, only a rule-of-thumb; in general, the best choice of bandwidth is data-specific. The kernel density estimate at a given point is a data-driven estimate of the value of the probability density function of the generating data distribution. Good results have been shown in [184] with a density-estimation method proposed in [316]. One should exclude the test point from the summation of Equation 4.9 to avoid overfitting, which is particularly important with certain types of kernel functions.

One can view the kernel density estimate as a simple non-parametric alternative to the fit values computed by the expectation-maximization algorithm in Chapter 2. Therefore, the kernel density estimates can be viewed as outlier scores in which smaller values indicate a greater degree of outlierness. The data points with unusually low density are declared outliers with the use of a Student's t-distribution or normal distribution assumption. However, the logarithmic function should be applied to the scores before extreme-value analysis, because such analysis is more effective on the log-likelihood scores.

Density-based methods face similar implementation challenges as histogram techniques. Just as the choice of grid-width causes a dilemma in histogram methods, the proper choice of bandwidth in kernel density methods is often data-distribution specific, which is not known a priori. Furthermore, the use of a global bandwidth in order to estimate density may not work very well in cases where there are wide variations in local density such as Figures 4.2 and 4.6. In a sense, the bandwidth plays a similar role in kernel-density estimation as the grid width in histogram methods, because both regulate the locality size of the estimation process. Furthermore, these methods are not very effective for higher dimensionality, because the accuracy of the density estimation process degrades with increasing dimensionality.

4.4.4.1 Connection with Harmonic k-Nearest Neighbor Detector

The harmonic k-nearest neighbor detector of Definition 4.3.3 is a special case of density-based methods. Specifically, the kernel function is simply set to the inverse distance to the

target. We replicate Equation 4.9 here:

$$\hat{f}(\overline{X}) = \frac{1}{N} \cdot \sum_{i=1}^{N} K'_h(\overline{X} - \overline{X_i})$$
$$\propto \frac{1}{N} \cdot \sum_{i=1}^{N} 1/||\overline{X} - \overline{X_i}||$$

Note that the second expression is simply the inverse of the harmonic score when the value of k is set to N in Definition 4.3.3. It is important to remove any training point $\overline{X_i}$ that is identical to $\overline{X}$ to avoid infinity scores caused by overfitting. Note that this kernel function does not use the bandwidth, and it is therefore a parameter-free detector. One can also use only the nearest $k < N$ points for the estimation in lieu of the bandwidth, although reasonably good results are often obtained at $k = N$. It is noteworthy that an (arithmetically) averaged k-nearest neighbor detector cannot discover isolated points near the center of the data distribution when k is set to N. For example, a single outlier at the center of a ring of points would receive the most *inlier-like* score in an average N-nearest neighbor detector. However, a harmonic N-nearest neighbor detector will do an excellent job in scoring the outlier appropriately, particularly if the value of N is large. This is because kernel densities are always estimated more accurately with larger data sets.

With an increasing number of points, this approach will indeed become a good *heuristic* estimate of the relative density. However, in many cases, one can obtain better results by repeatedly estimating $\hat{f}(\overline{X})$ with different samples of the data and averaging the scores to create an ensemble method. As in the case of all density-based methods, a logarithm function should be applied to $\hat{f}(\overline{X})$ (to convert it into a log-likelihood) before applying extreme-value analysis or ensemble methods.

4.4.4.2 Local Variations of Kernel Methods

It is possible to make kernel methods more locality sensitive by using a variety of tricks either in isolation or in combination:

1. The bandwidth for estimating the density at a point can be computed *locally* by using its k-nearest neighbor distance rather than using a global value.

2. The kernel density at a point can be normalized using the average kernel densities at its neighboring points.

These two tricks were used in combination in [342]. In addition, the distance in the exponent of the kernel function was replaced with the reachability distance (as defined for the LOF algorithm).

4.4.5 Ensemble-Based Implementations of Histograms and Kernel Methods

Histogram- and kernel-based techniques can be significantly strengthened using ensemble methods. The main problem in histogram-based techniques is to use them effectively in the context of high-dimensional data. In this context, it is possible to design effective implementations for high-dimensional data with the use of *rotated bagging* [32], because rotated bagging drastically reduces the dimensionality of the data from $O(d)$ to $O(\sqrt{d})$. The basic

idea is to generate a randomly rotated subspace of dimensionality $2+\lceil\sqrt{d}/2\rceil$ and project the data points in this subspace before applying a multidimensional histogram-based method (or any other detector). One might choose even less than $2+\lceil\sqrt{d}/2\rceil$ dimensions to ameliorate the effects of high dimensions. The specific details of the rotation process are described in section 5.3.3 of Chapter 5 and in section 6.4.4 of Chapter 6.

This approach is repeated multiple times with different random projections, and the scores of a point over different ensemble components are averaged in order to compute the final outlier score. The natural ensemble-based approach of rotated bagging implicitly constructs a more complex model than the simple histogram constructed by the base detectors. As a result, accuracy can be greatly improved. Note that rotated bagging was designed [32] as a meta-algorithm for *any* base detector and not just histogram-based methods.

A related method, referred to as *LODA* [436], shows how one can use 1-dimensional random projections with $\sqrt{d}$ non-zero elements in order to score the data points effectively. The histograms are built on 1-dimensional projections containing vectors of $\sqrt{d}$ non-zero elements, each of which is drawn from a standard normal distribution. The histograms are used to compute a log-likelihood score of each data point because it is easy to obtain probabilistic scores from histogram- and kernel density-based methods. The log-likelihood scores of each point from the different ensemble components are averaged in order to compute the final outlier score. The basic idea here is that even though 1-dimensional histograms are very weak detectors, it is often possible to combine such weak detectors in an ensemble in order to create a very strong outlier detector. The *LODA* method can be viewed as a special case of rotated bagging [32] in which 1-dimensional projections are combined with histogram-based detectors in order to obtain high-quality results. The *RS-Hash* method samples axis-parallel subspaces of varying dimensionality to score data points as outliers. When histogram and kernel methods are used in the ensemble-centric setting, it is important to apply the logarithm function to the estimated density before averaging the scores.

4.5 Limitations of Proximity-Based Detection

Most proximity-based methods use distances in order to define outliers at varying levels of granularity. Typically, higher levels of granularity are required for greater accuracy. In particular, methods that abstract the data by various forms of summarization do not distinguish well between true anomalies and noisy regions of low density. This weakness is manifested in the high variability in the prediction with different types of summarizations. In such cases, the variabilities in the predictions arising from the vagaries of the summarization process need to be ameliorated by ensemble methods. Furthermore, these methods need to combine global and local analysis carefully in order to find the true outliers in the data. A fully global analysis may miss important outliers as indicated in Figures 4.1 and 4.6, whereas a fully local analysis may miss small clustered groups of outliers as illustrated in Figure 4.2. At the same time, increasing the granularity of analysis can make the algorithms inefficient. In the worst-case, a distance-based algorithm with full granularity can require $O(N^2)$ distance computations in a data set containing N records. While indexing methods can be used in order to incorporate pruning into the outlier search, the effectiveness of pruning methods reduces with increasing dimensionality because of data sparsity.

An even more fundamental limitation in the context of high dimensional data is not one of *efficiency*, but that of the *quality* of the outliers found. In the high-dimensional case, all points become almost equidistant from one another, and therefore the contrast in the distances is lost [25, 263]. This is also referred to as the curse of dimensionality, which arises

from data sparsity, and it negatively impacts many high dimensional applications [8]. With increasing dimensionality, most of the features are not informative for outlier detection, and the noise effects of these dimensions will impact proximity-based methods in a very negative way. In such cases, the outliers can be masked by the noise in the features, unless the relevant dimensions can be explicitly discovered by an outlier detection method. Since proximity-based methods are naturally designed to use all the features in the data, their quality will naturally degrade with increasing dimensionality. Some methods do exist for improving the effectiveness of such methods in increasing dimensionality with subspace methods. These methods will be discussed in Chapter 5.

4.6 Conclusions and Summary

This chapter provides an overview of the key proximity-based techniques for outlier analysis. All these methods determine the outliers in the data with the use of proximity information between data points. These methods are closely related to clustering techniques in a complementary sense; while the former finds outlier points in sparse data localities, the latter tries to determine dense data localities. Therefore, clustering is itself a common method used in proximity-based outlier analysis. With the right choice of clustering method, it is also possible to make the approach adapt to patterns of arbitrary shape in the data. Such methods can also be extended to arbitrary data types.

Proximity-based methods enjoy wide popularity in the literature because of ease of implementation and interpretability. A major challenge in using such methods is that they are typically computationally intensive, and most of the high-quality methods require $O(N^2)$ distance computations in the worst-case. Various pruning methods can be used to improve their efficiency when binary labels are required instead of scores. Among the various pruning methods, sampling methods are among the most effective ones. Local normalization is a principle used to improve the effectiveness of various outlier detection algorithms. Two common local algorithms include the LOF and LOCI algorithms. Finally, histogram-based and kernel density estimation techniques have also been explored in literature. Although these methods have not found much popularity, they have significant potential when used in combination with ensemble methods.

4.7 Bibliographic Survey

The traditional definition of multivariate outliers often assumed them to be side products of clustering algorithms. Outliers were therefore defined as data points that do not naturally fit into any cluster. However, the non-membership of a data point in a cluster is not able to distinguish between noise and anomalies. A detailed discussion of different clustering algorithms and their use for outlier analysis may be found in [283, 307]. Many of the clustering algorithms explicitly label points that do not fit within clusters as outliers [594]. However, in some sparse high-dimensional domains, such as transaction data, (subspace)-clustering are among the few feasible methods for identifying outliers [254].

In order to improve the quality of clustering-based outlier scores, one can use the distance of data points to cluster centroids, rather than using only the membership of data points in clusters. The work in [499] investigates a number of deterministic and probabilistic methods for clustering in order to detect anomalies. These techniques were designed in the context of intrusion detection. One challenge in such methods is to prevent the clustering methods from quality-degradation from noise and anomalies already present in the data. This is

because if the discovered clusters are already biased by noise and anomalies, it will also prevent the effective identification of outliers. Such techniques have been used often in the context of intrusion-detection applications [70, 499]. The work in [70] uses a first phase in which the normal data is identified, by using data points matching frequent patterns in the data. These normal data are subsequently used in order to perform robust clustering. The outliers are then determined as points which lie at a significant distance to these clusters. These methods can be considered a kind of sequential ensemble approach.

A number of outlier detection methods have also been proposed for cases where the anomalies may lie in small clusters [188, 253, 291, 420, 445, 446]. Many of these techniques work by using distance-thresholding in order to regulate the creation of new clusters. When a data point does not lie within a specified threshold distance of the nearest cluster centroid, a new cluster is created containing a single instance. This results in clusters of varying size, since some of the newly created clusters do not get a sufficient number of points added to them. Then, the outlierness of a data point may be decided both by the number of points in its cluster, and the distance of its cluster to the other clusters. A number of indexing techniques have also been proposed in order to speed to the partitioning of the data points into clusters [131, 518]. Biased sampling [321] has also been shown to be an effective and efficient method for clustering-based outlier detection. A facility-location model for integrated clustering and outlier detection is proposed in [422]. The work in [475] shows how to construct robust spectral embeddings for outlier detection. Many of these methods can also be extended to arbitrary data types.

Distance-based methods have been extremely popular in the literature because of their ability to perform the analysis at a higher level of granularity than clustering methods. Furthermore, such methods are intuitive and extremely easy to understand and implement. The first distance-based method was proposed in [317]. The ideas in this work were extended to finding intensional knowledge in [318]. Subsequently, indexing methods were designed to improve the efficiency of this method in [456]. The work in [58] uses linearization in which the multidimensional space is populated with Hilbert space-filling curves. This 1-d representation has the advantage that the k-nearest neighbors can be determined very *fast* by examining the predecessors and successors of a data point on the space-filling curve. The sum of the k-nearest neighbor distances on the linearized representation is used in order to generate the outlier score of a data object. While the use of the *sum* of the k-nearest neighbor distances has some advantages over the k-nearest neighbor distance in differentiating between sparsely populated data and clustered data, it has the disadvantage of (sometimes) not being able to detect groups of isolated anomalies as illustrated in Figure 4.2. One challenge with the use of space-filling curves is that they map the data into a hypercube with d dimensions, and the number of corners of this hypercube increases exponentially with d. In such cases, the sparsity of the data in high dimensions may result in a degradation of the locality behavior of the space-filling curve. In order to address this issue, the work in [58] uses data-shifting techniques in order to improve locality. An iterative technique was designed, which requires $d + 1$ scans of the data set.

The work in [75] designs a simple sampling-based pruning technique in order to improve the efficiency of a k-nearest neighbor-based outlier detection technique. The core idea is similar to the pruning rule used in [456]. The idea is that if the outlier score for an object is less than the k-nearest neighbor distance of the rth best outlier, that data point cannot possibly be an outlier and is pruned from further consideration. This simple pruning rule has been shown in [75] to work well with randomized data. The randomization itself can be done in linear time by using a disk-based shuffling technique. The work in [572] performs the nearest neighbor computations on a smaller sample of the data set in order to improve

the efficiency. Theoretical guarantees are provided in order to bound the loss in accuracy resulting from the sampling process.

The effectiveness of pruning methods is clearly dependent on the ability to generate a good bound on the k-nearest neighbor distances in an efficient way. Therefore, the work in [219] partitions the data into small clusters. The k-nearest neighbor distance of a data point within a cluster is used in order to generate an upper bound on the k-nearest neighbor distance of that point. If this upper bound is less than the scores of the set of outliers already found, then the point can be pruned from consideration. The work in [456] also uses clustering techniques for pruning. The method in [219] uses recursive hierarchical partitioning in order to ensure that each cluster is assigned a similar number of data points. The ordering of the data points along the principal component of largest variance is used in order to provide a quick estimate of the k-nearest neighbor distance. In some cases, more accurate outlier detection can be performed by using a data-dependent similarity measure in k-nearest neighbor methods.

A discussion of data-dependent similarity functions is provided in [33, 491, 548]. Methods for data-dependent similarity include the shared-nearest neighbor measure [287], the pairwise global/local Mahalanobis adaptations (cf. Chapter 3 [33]), random forests [99, 491], randomized clustering trees [401, 555], and isolation forests [548]. It is noteworthy that isolation forests can be viewed as adaptations of extremely randomized clustering forests (ERC-Forests) [401] with a single trial at each node and growing the tree to full height with singleton points at the leaves.

A resolution-based method was proposed in [190]. According to this method, whether a point belongs to a cluster or whether it is an outlier depends on the distance threshold. At the highest resolution level, all points are in individual clusters of their own and are therefore outliers. As the resolution is slowly reduced, more and more data points join clusters. In each step, each point changes from being an outlier to a cluster. Based on this, a *Resolution Outlier Factor (ROF)* value was defined in [190]. This was shown to provide effective results for outlier analysis.

Most of the distance-based algorithms are designed with the use of Euclidean distances. In practice, the Euclidean function may not be optimal for finding the outliers. In fact, for many other domains of data, the distance functions are often defined in a fairly complex way, and many of the pruning techniques designed for Euclidean spaces will not work well in arbitrary spaces. In this context, an efficient algorithm was designed for outlier detection in arbitrary metric spaces [533], which requires at most three passes over the data.

A method to improve the efficiency of distance-based algorithms with the use of reference points was proposed in [430]. The core idea in this work is to rank the data points on the basis of their relative degree of density with respect to a fixed set of R reference points. Each data point is transformed to a 1-dimensional space in R possible ways on the basis of their distance to a reference point. For each of these R 1-dimensional data sets, the relative degree of density of each data point with respect to the corresponding reference point is computed. The overall relative degree of density of a data point is defined as the minimum relative degree of density over all the reference points. This relative degree of density provides a way to rank the different data points. Distributed algorithms for speeding up outlier detection are proposed in [83].

Scalability is a significant issue in the context of the data streams. Typically, in the case of data streams, a past window of history is used in order to determine outliers. Data points whose k-nearest neighbor values are large in a specific sliding window history are declared outliers [60, 322]. Stream-clustering methods such as those in [28] can also be used in order to speed up the outlier analysis process. Such an approach has been discussed in [322].

The issue of local density in the context of outlier analysis was first addressed in [96, 97]. We note that the reverse nearest neighbor approach [100, 248] presented in this chapter shares some similarities to LOF in terms of adjusting to local densities with the use of a reverse nearest-neighbor approach. The variations in local density may result in poor ranking of outliers by global distance-based methods. Therefore, the concept of Local Outlier Factor (LOF) was proposed in [96]. These methods adjust the outlier score of an object on the basis of the local density. It should be mentioned that the concept of density is really loosely defined in LOF as an inverse of averaged distances. A true definition of density should really count the number of data points in a specific volume. Data points in local regions of high density are given a higher outlier score, even if they are slightly isolated from the other points in their locality. A comparison of the LOF algorithm with respect to the average k-nearest neighbor algorithm is provided in [32]. It is shown that the average k-nearest neighbor algorithm is more robust.

Many different variants of the broad LOF approach were subsequently proposed. For example, the work in [293] proposed the concept of top-n local outliers, where the top-n outliers were determined on the basis of the density. Pruning techniques were used to improve the running time by partitioning the data into clusters, and computing bounds on the LOF values of the points in each cluster. Thus, entire clusters can be pruned if they are guaranteed to contain only points that have lower LOF values than the weakest of the current top-n outliers. Other methods for improving the effectiveness of top-n local outlier detection with the use of cluster-pruning were proposed in [144].

One issue with LOF is that it can sometimes be ineffective when regions of different density are not clearly separated. Therefore, the INFLO technique of [294] proposed a modification of LOF, which uses a symmetric nearest-neighbor relationship based on the nearest-neighbor distance as well as the reverse nearest-neighbor distance to order to define the local outliers. The concept of *connectivity-based outlier factor (COF)* was proposed in [534], which is also able to find outliers in low density or arbitrarily shaped regions effectively. The main difference between COF and LOF is the way in which COF defines the neighborhood of a data point. Specifically, the neighborhood is defined incrementally by adding the closest point to the current neighborhood set. This approach is based on a similar motivation as single-linkage clustering, because it merges the neighborhood-set (viewed as a cluster) and points (viewed as singleton clusters) based on exactly the same criterion as single-linkage clustering. Therefore, it can define arbitrarily shaped neighborhoods effectively when the points are distributed on arbitrary lower dimensional manifolds of the data. The LOF approach has also been combined with other clustering techniques. For example, the work in [253, 255] defines a score called *Cluster-Based Local Outlier Factor (CBLOF)* in which anomalies are defined as a combination of local distances to nearby clusters and the size of the clusters to which the data point belongs. Data points in small clusters that are at a large distance to nearby clusters are flagged as outliers.

The LOF scheme has also been extended to the case of spatial data with non-spatial attributes [523]. For example, the sea-surface temperature is a non-spatial attribute in the context of spatial location. Such data are known to exhibit *spatial auto-correlations*, in which the value of an element is affected by its immediate neighbors (e.g., spatial temperature locality). Furthermore, the data shows *spatial heteroscedasticity*, in which the variance of a data point is based on its location. For example, "normal" temperature variations are clearly based on geographical location. We note that spatial data share some similarities with temporal data from the perspective of *spatial continuity*, which is analogous to *temporal continuity*. Correspondingly, the work in [523] defines a local outlier measure, known as the *Spatial Local Outlier Measure (SLOM)*, which is specially suited to spatial outlier detection.

The generalization of the LOF method to the streaming scenario is discussed in [443].

The LOCI method [426] is also a locally sensitive method, which uses the number of points in a circular neighborhood around a point, rather than the inverse of the k-nearest neighbor distances for local density computation. Thus, it is truly a density-based method from an intuitive perspective. Furthermore, the approach is tested over different levels of granularity in order to reduce the parameter choices and remove the need for *some* of the input parameters during the outlier detection process. An approximate version of the algorithm can be implemented in almost linear time. An interesting contribution of this work is the introduction of LOCI plots, which provide an intuitive understanding of the outliers in the data with a visual plot. The LOCI plot provides an understanding of how different sizes of neighborhoods may correspond to the outlier score of a data point.

The traditional methods for density-based outlier analysis involve the use of discretization, grid-based methods, and kernel-density based methods. The first two belong in the general category of histogram-based methods [260, 288]. Histogram-based techniques find wide applicability in intrusion-detection techniques, in which it is natural to construct frequency-based profiles of various events. The main challenge in histogram-based methods is that the bucket-size along each dimension can sometimes be hard to pick correctly. The work in [476] proposes the *RS-Hash* method to perform subspace outlier detection in linear time by randomly varying the sizes and dimensions of the different grid regions in an ensemble-centric approach. The approach has also been extended to data streams in the same work. A closely related method is that of kernel-density estimation [262, 496]. Local variations of this approach are discussed in [342, 483]. Kernel density-estimation is a continuous variation of grid-based methods, in which a smooth kernel function is used for the estimation process. A particularly robust form of kernel density estimation, which has been shown [184] to achieve good results for anomaly detection, is provided in [316].

4.8 Exercises

1. Consider a data set with the following observations: { (1, 3), (1.01, 3.01), (0.99, 3.01), (0.99, 3), (0.99, 2.99), (3, 1) }.

- What are the results of linear modeling on this data set with 1-dimensional PCA for finding outliers?
- How well does a 1-NN technique work for finding outliers in this case?
- Discuss the main challenge in using PCA in this case. Is there any way that you can improve the scores from PCA?

2. Consider a data set containing a single cluster with the points { (1, 1), (0, 0), (2, 2.1), (3, 3.1), (4, 4), (5.1, 5) }.

- Consider the two points $(6.5, 6.5)$, and $(1, 2.1)$. Draw the points on a piece of paper. Which of the two data points seems more like an outlier?
- Which point does a 1-NN algorithm set as the highest outlier score with the Euclidean metric?
- Which point does a 1-NN algorithm set as the lowest outlier score with the Euclidean metric?
- Which data point does a rank-1 PCA-based algorithm set at the highest outlier rank, when the residuals are the outlier scores?

- Would you recommend changing the distance function from the Euclidean metric? How?

3. Download the Ionosphere data set from the UCI Machine Learning Repository [203].

- Rank the data points based on their residual scores in a PCA approach, when only the top 3 eigenvectors are used.
- Rank the data points based on their k-nearest neighbor scores, for values of k ranging from 1 through 5.
- Normalize the data, so that the variance along each dimension is 1. Rank the data points based on their k-nearest neighbor scores, for values of k ranging from 1 through 5.
- How many data points are common among the top 5 ranked outliers using different methods?
- Now use a voting scheme, which adds up the ranks of the outliers in different schemes. Which are the top 5 outliers now?
- Does this ensemble approach provide more robust outliers?

4. Repeat Exercise 3 with the network intrusion data set from the UCI Machine Learning Repository.

5. A manufacturing company produces 2-dimensional square widgets, which are normally distributed with a length of 1 meter on each side, and a standard deviation of 0.01 meters.

- Generate a data set with 100,000 widgets from this distribution.
- The company produced 5 anomalous widgets, due a defect in the manufacturing process. Each such widget had a square length of 0.1 meters, and standard deviation of 0.001 meters. Generate these 5 anomalous points using the normal distribution assumption.
- Does a 1-NN approach find the anomalous widgets?
- Does a 10-NN approach find the anomalous widgets?

6. Apply a k-means clustering approach to the data set in Exercise 5, where 5 cluster centroids are used. As a post-processing step, remove any clusters with 10 or less data points. Score the data points by their distance to their closest cluster centroids. Which data points have the highest outlier scores?

7. Apply the reverse 1-NN algorithm to the case of Exercise 5. Which data points have the highest outlier scores? Which data points have the highest outlier scores with the reverse 10-NN algorithm? With the reverse 100-NN algorithm?

8. Repeat Exercises 3 and 4 with the use of the LOF method and determine the ranking of the outliers. Are the outliers same in this case as those found in Exercises 3 and 4?

9. Repeat Exercise 8 with the use of the LOCI method. Are the outliers found to be the same?