Advancements of Outlier Detection: A Survey

Ji Zhang

Department of Mathematics and Computing
University of Southern Queensland, Australia

Abstract

Outlier detection is an important research problem in data mining that aims to discover useful abnormal and irregular patterns hidden in large datasets. In this paper, we present a survey of outlier detection techniques to reflect the recent advancements in this field. The survey will not only cover the traditional outlier detection methods for static and low dimensional datasets but also review the more recent developments that deal with more complex outlier detection problems for dynamic/streaming and high-dimensional datasets.

Keywords: Data Mining, Outlier Detection, High-dimensional Datasets

Received on 29 December 2011; accepted on 18 February 2012; published on 04 February 2013

doi:10.4108/trans.sis.2013.01-03.e2

1. Introduction

Outlier detection is an important research problem in data mining that aims to find objects that are considerably dissimilar, exceptional and inconsistent with respect to the majority data in an input database [50]. Outlier detection, also known as anomaly detection in some literatures, has become the enabling underlying technology for a wide range of practical applications in industry, business, security and engineering, etc. For example, outlier detection can help identify suspicious fraudulent transaction for credit card companies. It can also be utilized to identify abnormal brain signals that may indicate the early development of brain cancers. Due to its inherent importance in various areas, considerable research efforts in outlier detection have been conducted in the past decade. A number of outlier detection techniques have been proposed that use different mechanisms and algorithms. This paper presents a comprehensive review on the major state-of-the-art outlier detection methods. We will cover different major categories of outlier detection approaches and critically evaluate their respective advantages and disadvantages.

In principle, an outlier detection technique can be considered as a mapping function f that can be expressed as $f(p) \rightarrow q$, where $q \in \Re^+$. Giving a data point p in the given dataset, a corresponding outlier-ness score is generated by applying the mapping function f to quantitatively reflect the strength of outlier-ness of p. Based on the mapping function f, there are typically two major tasks for outlier detection problem to accomplish, which leads to two corresponding problem formulations. From the given dataset that is under study, one may want to find the top k outliers that have the highest outlier-ness scores or all the outliers whose outlier-ness score exceeding a user specified threshold.

The exact techniques or algorithms used in different outlier methods may vary significantly, which are largely dependent on the characteristic of the datasets to be dealt with. The datasets could be static with a small number of attributes where outlier detection is relatively easy. Nevertheless, the datasets could also be dynamic, such as data streams, and at the same time have a large number of attributes. Dealing with this kind of datasets is more complex by nature and requires special attentions to the detection performance (including speed and accuracy) of the methods to be developed.

Given the abundance of research literatures in the field of outlier detection, the scope of this survey will be clearly specified first in order to facilitate a systematic survey of the existing outlier detection methods. After that, we will start the survey with a review of the conventional outlier detection techniques that are primarily suitable for relatively low-dimensional

*Corresponding author. Email: Ji.Zhang@usq.edu.au static data, followed by some of the major recent advancements in outlier detection for high-dimensional static data and data streams.

2. Scope of This Survey

Before the review of outlier detection methods is presented, it is necessary for us to first explicitly specify the scope of this survey. There have been a lot of research work in detecting different kinds of outliers from various types of data where the techniques outlier detection methods utilize differ considerably. Most of the existing outlier detection methods detect the so-called *point outliers from vector-like data sets*. This is the focus of this review as well as of this thesis. Another common category of outliers that has been investigated is called *collective outliers*. Besides the vector-like data, outliers can also be detected from other types of data such as sequences, trajectories and graphs, etc. In the reminder of this subsection, we will discuss briefly different types of outliers.

First, outliers can be classified as point outliers and collective outliers based on the number of data instances involved in the concept of outliers.

- **Point outliers.** In a given set of data instances, an individual outlying instance is termed as a point outlier. This is the simplest type of outliers and is the focus of majority of existing outlier detection schemes [28]. A data point is detected as a point outlier because it displays outlier-ness at its own right, rather than together with other data points. In most cases, data are represented in vectors as in the relational databases. Each tuple contains a specific number of attributes. The principled method for detecting point outliers from vector-type data sets is to quantify, through some outlier-ness metrics, the extent to which each single data is deviated from the other data in the data set.

- **Collective outliers.** A collective outlier represents a collection of data instances that is outlying with respect to the entire data set. The individual data instance in a collective outlier may not be outlier by itself, but the joint occurrence as a collection is anomalous [28]. Usually, the data instances in a collective outlier are related to each other. A typical type of collective outliers are sequence outliers, where the data are in the format of an ordered sequence.

Outliers can also be categorized into vector outliers, sequence outliers, trajectory outliers and graph outliers, etc, depending on the types of data from where outliers can be detected.

- **Vector outliers.** Vector outliers are detected from vector-like representation of data such as the relational databases. The data are presented in tuples and each tuple has a set of associated attributes. The data set can contain only numeric attributes, or categorical attributes or both. Based on the number of attributes, the data set can be broadly classified as low-dimensional data and high-dimensional data, even though there is not a clear cutoff between these two types of data sets. As relational databases still represent the mainstream approaches for data storage, therefore, vector outliers are the most common type of outliers we are dealing with.

- **Sequence outliers.** In many applications, data are presented as a sequence. A good example of a sequence database is the computer system call log where the computer commands executed, in a certain order, are stored. A sequence of commands in this log may look like the following sequence: *http-web*, *buffer-overflow*, *http-web*, *http-web*, *smtp-mail*, *ftp*, *http-web*, *ssh*. Outlying sequence of commands may indicate a malicious behavior that potentially compromises system security. In order to detect abnormal command sequences, normal command sequences are maintained and those sequences that do not match any normal sequences are labeled sequence outliers. Sequence outliers are a form of collective outlier.

- **Trajectory outliers.** Recent improvements in satellites and tracking facilities have made it possible to collect a huge amount of trajectory data of moving objects. Examples include vehicle positioning data, hurricane tracking data, and animal movement data [65]. Unlike a vector or a sequence, a trajectory is typically represented by a set of key features for its movement, including the coordinates of the starting and ending points; the average, minimum, and maximum values of the directional vector; and the average, minimum, and maximum velocities. Based on this representation, a weighted-sum distance function can be defined to compute the difference of trajectory based on the key features for the trajectory [60]. A more recent work proposed a partition-and-detect framework for detecting trajectory outliers [65]. The idea of this method is that it partitions the whole trajectory into line segments and tries to detect outlying line segments, rather than the whole trajectory. Trajectory outliers can be point outliers if we consider each single trajectory as the basic data unit in the outlier detection. However, if the moving objects in the trajectory are considered, then an abnormal sequence of such moving objects (constituting the sub-trajectory) is a collective outlier.

- **Graph outliers.** Graph outliers represent those graph entities that are abnormal when compared with their peers. The graph entities that can become outliers include nodes, edges and sub-graphs. For example, Sun *et al.* investigate the detection of anomalous nodes in a bipartite graph [84][85]. Autopart detects outlier edges in a general graph [27]. Noble *et al.* study anomaly detection on a general graph with labeled nodes and try to identify abnormal substructure in the graph [72]. Graph outliers can be either point outliers (*e.g.*, node and edge outliers) or collective outliers (*e.g.*, sub-graph outliers).

Unless otherwise stated, all the outlier detection methods discussed in this review refer to those methods for detecting point outliers from vector-like data sets.

3. Related Work

4. Outlier Detection Methods for Low Dimensional Data

The earlier research work in outlier detection mainly deals with static datasets with relatively low dimensions. Literature on these work can be broadly classified into four major categories based on the techniques they used, *i.e.*, statistical methods, distance-based methods, density-based methods and clustering-based methods.

4.1. Statistical Detection Methods

Statistical outlier detection methods [23, 47] rely on the statistical approaches that assume a distribution or probability model to fit the given dataset. Under the distribution assumed to fit the dataset, the outliers are those points that do not agree with or conform to the underlying model of the data.

The statistical outlier detection methods can be broadly classified into two categories, *i.e.*, the parametric methods and the non-parametric methods. The major differences between these two classes of methods lie in that the parametric methods assume the underlying distribution of the given data and estimate the parameters of the distribution model from the given data [34] while the non-parametric methods do not assume any knowledge of distribution characteristics [31].

Statistical outlier detection methods (parametric and non-parametric) typically take two stages for detecting outliers, *i.e.*, the training stage and test stage.

- **Training stage**. The training stage mainly involves fitting a statistical model or building data profiles based on the given data. Statistical techniques can be performed in a supervised, semi-supervised, and unsupervised manner. Supervised techniques estimate the probability density for normal instances and outliers. Semi-supervised techniques estimate the probability density for either normal instances, or outliers, depending on the availability of labels. Unsupervised techniques determine a statistical model or profile which fits all or the majority of the instances in the given data set;

- **Test stage**. Once the probabilistic model or profile is constructed, the next step is to determine if a given data instance is an outlier with respect to the model/profile or not. This involves computing the posterior probability of the test instance to be generated by the constructed model or the deviation from the constructed data profile. For example, we can find the distance of the data instance from the estimated mean and declare any point above a threshold to be an outlier [42].

Parametric Methods. Parametric statistical outlier detection methods explicitly assume the probabilistic or distribution model(s) for the given data set. Model parameters can be estimated using the training data based upon the distribution assumption. The major parametric outlier detection methods include Gaussian model-based and regression model-based methods.

A. Gaussian Models

Detecting outliers based on Gaussian distribution models have been intensively studied. The training stage typically performs estimation of the mean and variance (or standard deviation) of the Gaussian distribution using Maximum Likelihood Estimates (MLE). To ensure that the distribution assumed by human users is the optimal or close-to-optima underlying distribution the data fit, statistical discordany tests are normally conducted in the test stage [23][16][18]. So far, over one hundred discordancy/outlier tests have been developed for different circumstances, depending on the parameter of dataset (such as the assumed data distribution) and parameter of distribution (such as mean and variance), and the expected number of outliers [50][58]. The rationale is that some small portion of points that have small probability of occurrence in the population are identified as outliers. The commonly used outlier tests for normal distributions are the *mean-variance test* and *box-plot test* [66][49][83][44]. In the mean-variance test for a Gaussian distribution $N(\mu, \sigma^2)$, where the population has a mean μ and variance σ, outliers can be considered to be points that lie 3 or more standard deviations (*i.e.*, $\geq 3\sigma$) away from the mean [41]. This test is general and can be applied to some other commonly used distributions such as Student t-distribution and Poisson distribution, which feature a fatter tail and a longer right tail than a normal distribution, respectively. The box-plot test draws on the box plot to graphically depict the distribution of data using five major attributes, *i.e.*, smallest non-outlier observation (min), lower quartile

(Q1), median, upper quartile (Q3), and largest non-outlier observation (max). The quantity Q3-Q1 is called the *Inter Quartile Range (IQR)*. IQR provides a means to indicate the boundary beyond which the data will be labeled as outliers; a data instance will be labeled as an outlier if it is located 1.5*IQR times lower than Q1 or 1.5*IQR times higher than Q3.

In some cases, a mixture of probabilistic models may be used if a single model is not sufficient for the purpose of data modeling. If labeled data are available, two separate models can be constructed, one for the normal data and another for the outliers. The membership probability of the new instances can be quantified and they are labeled as outliers if their membership probability of outlier probability model is higher than that of the model of the normal data. The mixture of probabilistic models can also be applied to unlabeled data, that is, the whole training data are modeled using a mixture of models. A test instance is considered to be an outlier if it is found that it does not belong to any of the constructed models.

B. Regression Models

If the probabilistic model is unknown regression can be employed for model construction. The regression analysis aims to find a dependence of one/more random variable(s) $\mathcal{Y}$ on another one/more variable(s) $\mathcal{X}$. This involves examining the conditional probability distribution $\mathcal{Y}|\mathcal{X}$. Outlier detection using regression techniques are intensively applied to time-series data [4][2][39][1][64]. The training stage involves constructing a regression model that fits the data. The regression model can either be a linear or non-linear model, depending on the choice from users. The test stage tests the regression model by evaluating each data instance against the model. More specifically, such test involves comparing the actual instance value and its projected value produced by the regression model. A data point is labeled as an outlier if a remarkable deviation occurs between the actual value and its expected value produced by the regression model.

Basically speaking, there are two ways to use the data in the dataset for building the regression model for outlier detection, namely the *reverse search* and *direct search* methods. The reverse search method constructs the regression model by using all data available and then the data with the greatest error are considered as outliers and excluded from the model. The direct search approach constructs a model based on a portion of data and then adds new data points incrementally when the preliminary model construction has been finished. Then, the model is extended by adding most fitting data, which are those objects in the rest of the population that have the least deviations from the model constructed thus far. The data added to the model in the last round, considered to be the least fitting data, are regarded to be outliers.

Non-parametric Methods. The outlier detection techniques in this category do not make any assumptions about the statistical distribution of the data. The most popular approaches for outlier detection in this category are histograms and Kernel density function methods.

A. Histograms

The most popular non-parametric statistical technique is to use histograms to maintain a profile of data. Histogram techniques by nature are based on the frequency or counting of data.

The histogram based outlier detection approach is typically applied when the data has a single feature. Mathematically, a histogram for a feature of data consists of a number of disjoint bins (or buckets) and the data are mapped into one (and only one) bin. Represented graphically by the histogram graph, the height of bins corresponds to the number of observations that fall into the bins. Thus, if we let n be the total number of instances, k be the total number of bins and m_i be the number of data point in the i^{th} bin ($1 \leq i \leq k$), the histogram satisfies the following condition $n = \sum_{i=1}^{k} m_i$. The training stage involves building histograms based on the different values taken by that feature in the training data.

The histogram techniques typically define a measure between a new test instance and the histogram based profile to determine if it is an outlier or not. The measure is defined based on how the histogram is constructed in the first place. Specifically, there are three possible ways for building a histogram:

1. The histogram can be constructed only based on normal data. In this case, the histogram only represents the profile for normal data. The test stage evaluates whether the feature value in the test instance falls in any of the populated bins of the constructed histogram. If not, the test instance is labeled as an outlier [5] [54][48];

2. The histogram can be constructed only based on outliers. As such, the histogram captures the profile for outliers. A test instance that falls into one of the populated bins is labeled as an outlier [32]. Such techniques are particularly popular in intrusion detection community [34][38] [30] and fraud detection [40];

3. The histogram can be constructed based on a mixture of normal data and outliers. This is the typical case where histogram is constructed. Since normal data typically dominate the whole data set, thus the histogram represents an approximated profile of normal data. The sparsity

of a bin in the histogram can be defined as the ratio of frequency of this bin against the average frequency of all the bins in the histogram. A bin is considered as sparse if such ratio is lower than a user-specified threshold. All the data instance falling into the sparse bins are labeled as outliers.

The first and second ways for constructing histogram, as presented above, rely on the availability of labeled instances, while the third one does not.

For multivariate data, a common approach is to construct feature-wise histograms. In the test stage, the probability for each feature value of the test data is calculated and then aggregated to generate the so-called *outlier score*. A low probability value corresponds a higher outlier score of that test instance. The aggregation of per-feature likelihoods for calculating outlier score is typically done using the following equation:

$$Outlier\_Score = \sum_{f \in F} w_f \cdot (1 - p_f)/|F|$$

where w_f denotes the weight assigned for feature f, p_f denotes the probability for the value of feature f and F denotes the set of features of the dataset. Such histogram-based aggregation techniques have been used in intrusion detection in system call data [35], fraud detection [40], damage detection in structures [67] [70] [71], network intrusion detection [90] [91], web-based attack detection [63], Packet Header Anomaly Detection (PHAD), Application Layer Anomaly Detection (ALAD) [69], NIDES (by SRI International) [5] [12] [79]. Also, a substantial amount of research has been done in the field of outlier detection for sequential data (primarily to detect intrusions in computer system call data) using histogram based techniques. These techniques are fundamentally similar to the instance based histogram approaches as described above but are applied to sequential data to detect collective outliers.

Histogram based detection methods are simple to implement and hence are quite popular in domain such as intrusion detection. But one key shortcoming of such techniques for multivariate data is that they are not able to capture the interactions between different attributes. An outlier might have attribute values that are individually very frequent, but their combination is very rare. This shortcoming will become more salient when dimensionality of data is high. A feature-wise histogram technique will not be able to detect such kinds of outliers. Another challenge for such techniques is that users need to determine an optimal size of the bins to construct the histogram.

B. Kernel Functions

Another popular non-parametric approach for outlier detection is the parzen windows estimation due to Parzen [76]. This involves using Kernel functions to approximate the actual density distribution. A new instance which lies in the low probability area of this density is declared to be an outlier.

Formally, if $x_1, x_2, ..., x_N$ are IID (independently and identically distributed) samples of a random variable x, then the Kernel density approximation of its *probability density function (pdf)* is

$$f_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x - x_i}{h})$$

where K is Kernel function and h is the bandwidth (smoothing parameter). Quite often, K is taken to be a standard Gaussian function with mean $\mu = 0$ and variance $\sigma^2 = 1$:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

Novelty detection using Kernel function is presented by [17] for detecting novelties in oil flow data. A test instance is declared to be novel if it belongs to the low density area of the learnt density function. Similar application of parzen windows is proposed for network intrusion detection [29] and for mammographic image analysis [86]. A semi-supervised probabilistic approach is proposed to detect novelties [31]. Kernel functions are used to estimate the probability distribution function (pdf) for the normal instances. Recently, Kernel functions are used in outlier detection in sensor networks [80][25].

Kernel density estimation of pdf is applicable to both univariate and multivariate data. However, the pdf estimation for multivariate data is much more computationally expensive than the univariate data. This renders the Kernel density estimation methods rather inefficient in outlier detection for high-dimensional data.

Advantages and Disadvantages of Statistical Methods. Statistical outlier detection methods feature some advantages. They are mathematically justified and if a probabilistic model is given, the methods are very efficient and it is possible to reveal the meaning of the outliers found [75]. In addition, the model constructed, often presented in a compact form, makes it possible to detect outliers without storing the original datasets that are usually of large sizes.

However, the statistical outlier detection methods, particularly the parametric methods, suffer from some key drawbacks. First, they are typically not applied in a multi-dimensional scenario because most distribution models typically apply to the univariate feature space. Thus, they are unsuitable even for moderate multi-dimensional data sets. This greatly limits their applicability as in most practical applications the data is multiple or even high dimensional. In addition, a lack of the prior knowledge regarding the underlying distribution of the dataset makes the distribution-based methods difficult to use in practical applications. A single distribution may not model the entire data because the data may originate from multiple distributions. Finally, the quality of results cannot be guaranteed because they are largely dependent on the distribution chosen to fit the data. It is not guaranteed that the data being examined fit the assumed distribution if there is no estimate of the distribution density based on the empirical data. Constructing such tests for hypothesis verification in complex combinations of distributions is a nontrivial task whatsoever. Even if the model is properly chosen, finding the values of parameters requires complex procedures. From above discussion, we can see the statistical methods are rather limited to large real-world databases which typically have many different fields and it is not easy to characterize the multivariate distribution of exemplars.

For non-parametric statistical methods, such as histogram and Kernal function methods, they do not have the problem of distribution assumption that the parametric methods suffer and they both can deal with data streams containing continuously arriving data. However, they are not appropriate for handling high-dimensional data. Histogram methods are effective for a single feature analysis, but they lose much of their effectiveness for multi or high-dimensional data because they lack the ability to analyze multiple feature simultaneously. This prevents them from detecting subspace outliers. Kernel function methods are appropriate only for relatively low dimensional data as well. When the dimensionality of data is high, the density estimation using Kernel functions becomes rather computationally expensive, making it inappropriate for handling high-dimensional data streams.

4.2. Distance-based Methods

There have already been a number of different ways for defining outliers from the perspective of distance-related metrics. Most existing metrics used for distance-based outlier detection techniques are defined based upon the concepts of *local neighborhood* or k nearest neighbors (kNN) of the data points. The notion of distance-based outliers does not assume any underlying data distributions and generalizes many concepts from distribution-based methods. Moreover, distance-based methods scale better to multi-dimensional space and can be computed much more efficiently than the statistical-based methods.

In distance-based methods, distance between data points is needed to be computed. We can use any of the L_p metrics like the Manhattan distance or Euclidean distance metrics for measuring the distance between a pair of points. Alternately, for some other application domains with presence of categorical data (*e.g.*, text documents), non-metric distance functions can also be used, making the distance-based definition of outliers very general. Data normalization is normally carried out in order to normalize the different scales of data features before outlier detection is performed.

A. Local Neighborhood Methods

The first notion of distance-based outliers, called $DB(k, \lambda)$-Outlier, is due to Knorr and Ng [58]. It is defined as follows. A point p in a data set is a $DB(k, \lambda)$-Outlier, with respect to the parameters k and λ, if no more than k points in the data set are at a distance λ or less (*i.e.*, λ–neighborhood) from p. This definition of outliers is intuitively simple and straightforward. The major disadvantage of this method, however, is its sensitivity to the parameter λ that is difficult to specify a priori. As we know, when the data dimensionality increases, it becomes increasingly difficult to specify an appropriate circular local neighborhood (delimited by λ) for outlier-ness evaluation of each point since most of the points are likely to lie in a thin shell about any point [19]. Thus, a too small λ will cause the algorithm to detect all points as outliers, whereas no point will be detected as outliers if a too large λ is picked up. In other words, one needs to choose an appropriate λ with a very high degree of accuracy in order to find a modest number of points that can then be defined as outliers.

To facilitate the choice of parameter values, this first local neighborhood distance-based outlier definition is extended and the so-called $DB(pct, d_{min})$-Outlier is proposed which defines an object in a dataset as a $DB(pct, d_{min})$-Outlier if at least pct% of the objects in the datasets have the distance larger than d_{min} from this object [59][60]. Similar to $DB(k, \lambda)$-Outlier, this method essentially delimits the local neighborhood of data points using the parameter d_{min} and measures the outlierness of a data point based on the percentage, instead of the absolute number, of data points falling into this specified local neighborhood. As pointed out in [56] and [57], $DB(pct, d_{min})$ is quite general and is able to unify the exisiting statisical detection methods using discordancy tests for outlier detection. For exmaple, $DB(pct, d_{min})$ unifies the definition of outliers using a normal distribution-based discordancy test with $pct = 0.9988$ and $d_{min} = 0.13$. The specification of pct is obviously more intuitive and easier than the specification of k in $DB(k, \lambda)$-Outliers [59]. However, $DB(pct, d_{min})$-Outlier suffers a similar problem as $DB(pct, d_{min})$-Outlier in specifying the local neighborhood parameter d_{min}.

To efficiently calculate the number (or percentage) of data points falling into the local neighborhood of each point, three classes of algorithms have been presented, *i.e.*, the nested-loop, index-based and cell-based algorithms. For easy of presentation, these three algorithms are discussed for detecting $DB(k, \lambda)$-Outlier.

The *nested-loop algorithm* uses two nested loops to compute $DB(k, \lambda)$-Outlier. The outer loop considers each point in the dataset while the inner loop computes for each point in the outer loop the number (or percentage) of points in the dataset falling into the specified λ-neighborhood. This algorithm has the advantage that it does not require the indexing structure be constructed at all that may be rather expensive at most of the time, though it has a quadratic complexity with respect to the number of points in the dataset.

The *index-based algorithm* involves calculating the number of points belonging to the λ-neighborhood of each data by intensively using a pre-constructed multi-dimensional index structure such as R^*-tree [22] to facilitate kNN search. The complexity of the algorithm is approximately logarithmic with respect to the number of the data points in the dataset. However, the construction of index structures is sometimes very expensive and the quality of the index structure constructed is not easy to guarantee.

In the *cell-based algorithm*, the data space is partitioned into cells and all the data points are mapped into cells. By means of the cell size that is known a priori, estimates of pair-wise distance of data points are developed, whereby heuristics (pruning properties) are presented to achieve fast outlier detection. It is shown that three passes over the dataset are sufficient for constructing the desired partition. More precisely, the $d-$dimensional space is partitioned into cells with side length of $\frac{\lambda}{2\sqrt{d}}$. Thus, the distance between points in any 2 neighboring cells is guaranteed to be at most λ. As a result, if for a cell the total number of points in the cell and its neighbors is greater than k, then none of the points in the cell can be outliers. This property is used to eliminate the vast majority of points that cannot be outliers. Also, points belonging to cells that are more than 3 cells apart are more than a distance λ apart. As a result, if the number of points contained in all cells that are at most 3 cells away from the a given cell is less than k, then all points in the cell are definitely outliers. Finally, for those points that belong to a cell that cannot be categorized as either containing only outliers or only non-outliers, only points from neighboring cells that are at most 3 cells away need to be considered in order to determine whether or not they are outliers. Based on the above properties, the authors propose a three-pass algorithm for computing outliers in large databases. The time complexity of this cell-based algorithm is $O(c^d + N)$, where c is a number that is inversely proportional to λ. This complexity is linear with dataset size N but exponential with the number of dimensions d. As a result, due to the exponential growth in the number of cells as the number of dimensions is increased, the cell-based algorithm starts to perform poorly than the nested loop for datasets with dimensions of 4 or higher.

In [36], a similar definition of outlier is proposed. It calculates the number of points falling into the w-radius of each data point and labels those points as outliers that have low neighborhood density. We consider this definition of outliers as the same as that for $DB(k, \lambda)$-Outlier, differing only that this method does not present the threshold k explicitly in the definition. As the computation of the local density for each point is expensive, [36] proposes a clustering method for an efficient estimation. The basic idea of such approximation is to use the size of a cluster to approximate the local density of all the data in this cluster. It uses the fix-width clustering [36] for density estimation due to its good efficiency in dealing with large data sets.

B. kNN-distance Methods

There have also been a few distance-based outlier detection methods utilizing the k nearest neighbors (kNN) in measuring the outlier-ness of data points in the dataset. The first proposal uses the distance to the k^{th} nearest neighbors of every point, denoted as D^k, to rank points so that outliers can be more efficiently discovered and ranked [81]. Based on the notion of D^k, the following definition for D^k_n-Outlier is given: Given k and n, a point is an outlier if the distance to its k^{th} nearest neighbor of the point is smaller than the corresponding value for no more than $n-1$ other points. Essentially, this definition of outliers considers the top n objects having the highest D^k values in the dataset as outliers.

Similar to the computation of $DB(k, \lambda)$-Outlier, three different algorithms, *i.e.*, the nested-loop algorithm, the index-based algorithm, and the partition-based algorithm, are proposed to compute D^k for each data point efficiently.

The *nested-loop algorithm* for computing outliers simply computes, for each input point p, D^k, the distance of between p and its k^{th} nearest neighbor. It then sorts the data and selects the top n points with the maximum D^k values. In order to compute D^k for points, the algorithm scans the database for each point p. For a point p, a list of its k nearest points is maintained, and for each point q from the database which is considered, a check is made to see if the distance between p and q is smaller than the distance of the k^{th} nearest neighbor found so far. If so, q is included in the list of the k nearest neighbors for p. The moment that the list contains more than k neighbors, then the point that is furthest away from p is deleted from the list. In this algorithm, since only one point is processed at a time, the database would need to be scanned N times, where N is the number of points in the database. The computational complexity is in the order of $O(N^2)$, which is rather expensive for large datasets. However, since we are only interested in the top n outliers, we can apply the following pruning optimization to early-stop the computation of D^k for a point p. Assume that during each step of the algorithm, we store the top n outliers computed thus far. Let D^n_{min} be the minimum among these top n outliers. If during the computation of for a new point p, we find that the value for D^k computed so far has fallen below D^n_{min}, we are guaranteed that point p cannot be an outlier. Therefore, it can be safely discarded. This is because D^k monotonically decreases as we examine more points. Therefore, p is guaranteed not to be one of the top n outliers.

The *index-based algorithm* draws on index structure such as R*-tree [22] to speed up the computation. If we have all the points stored in a spatial index like R*-tree, the following pruning optimization can be applied to reduce the number of distance computations. Suppose that we have computed for point p by processing a portion of the input points. The value that we have is clearly an upper bound for the actual D^k of p. If the minimum distance between p and the *Minimum Bounding Rectangles* (MBR) of a node in the R*-tree exceeds the value that we have anytime in the algorithm, then we can claim that none of the points in the sub-tree rooted under the node will be among the k nearest neighbors of p. This optimization enables us to prune entire sub-trees that do not contain relevant points to the kNN search for p.

The major idea underlying the *partition-based algorithm* is to first partition the data space, and then prune partitions as soon as it can be determined that they cannot contain outliers. Partition-based algorithm is subject to the pre-processing step in which data space is split into cells and data partitions, together with the Minimum Bounding Rectangles of data partitions, are generated. Since n will typically be very small, this additional preprocessing step performed at the granularity of partitions rather than points is worthwhile as it can eliminate a significant number of points as outlier candidates. This partition-based algorithm takes the following four steps:

1. First, a clustering algorithm, such as BIRCH, is used to cluster the data and treat each cluster as a separate partition;

2. For each partition P, the lower and upper bounds (denoted as $P.lower$ and $P.upper$, respectively) on D^k for points in the partition are computed. For every point $p \in P$, we have $P.lower \leq D^k(p) \leq P.upper$;

3. The *candidate partitions*, the partitions containing points which are candidates for outliers, are identified. Suppose we could compute $minDkDist$, the lower bound on D^k for the n outliers we have detected so far. Then, if $P.upper < minDkDist$, none of the points in P can possibly be outliers and are safely pruned. Thus, only partitions P for which $P.upper \geq minDkDist$ are chosen as candidate partitions;

4. Finally, the outliers are computed from among the points in the candidate partitions obtained in Step 3. For each candidate partition P, let $P.neighbors$ denote the neighboring partitions of P, which are all the partitions within distance $P.upper$ from P. Points belonging to neighboring partitions of P are the only points that need to be examined when computing D^k for each point in P.

The D^k_n-Outlier is further extended by considering for each point the sum of its k nearest neighbors [10]. This extension is motivated by the fact that the definition of D^k merely considers the distance between an object with its k^{th} nearest neighbor, entirely ignoring the distances between this object and its another $k-1$ nearest neighbors. This drawback may make D^k fail to give an accurate measurement of outlier-ness of data points in some cases. For a better understanding, we present an example, as shown in Figure 1, in which the same D^k value is assigned to points p_1 and p_2, two points with apparently rather different outlier-ness. The $k-1$ nearest neighbors for p_2 are populated much more densely around it than those of p_1, thus the outlier-ness of p_2 is obviously lower than p_1. Obviously, D^k is not robust enough in this example to accurately reveal the outlier-ness of data points. By summing up the distances between the object with all of its k nearest neighbors, we will be able to have a more accurate measurement of outlier-ness of the object, though this will require more computational effort in summing up the distances. This method is also used in [36] for anomaly detection.

The idea of kNN-based distance metric can be extended to consider the k nearest dense regions. The recent methods are the Largest_cluster method [61][98] and Grid-ODF [89], as discussed below.

Khoshgoftaar *et al.* propose a distance-based method for labeling wireless network traffic records in the data stream used as either normal or intrusive [61][98]. Let d be the largest distance of an instance to the centriod of the largest cluster. Any instance or cluster that has a distance greater than αd $(\alpha \geq 1)$ to the largest cluster is defined as an attack. This method is referred to as the Largest_Cluster method. It can also be used to detect outliers. It takes the following several steps for outlier detection:

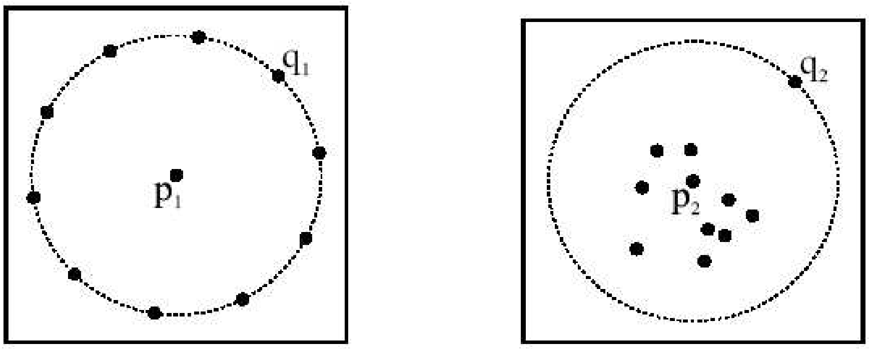


Figure 1. Points with the same D^k value but different outlier-ness

1. Find the largest cluster, *i.e.* the cluster with largest number of instances, and label it as normal. Let c_0 be the centriod of this cluster;

2. Sort the remaining clusters in ascending order based on the distance from their cluster centroid to c_0;

3. Label all the instances that have a distance to c_0 greater than αd, where α is a human-specified parameter;

4. Label all the other instances as normal.

When used in dealing with projected anomalies detection for high-dimensional data streams, this method suffers the following limitations:

- First and most importantly, this method does not take into account the nature of outliers in high-dimensional data sets and is unable to explore subspaces to detect projected outliers;

- k-means clustering is used in this method as the backbone enabling technique for detecting intrusions. This poses difficulty for this method to deal with data streams. k-means clustering requires iterative optimization of clustering centroids to gradually achieve better clustering results. This optimization process involves multiple data scans, which is infeasible in the context of data streams;

- A strong assumption is made in this method that all the normal data will appear in a single cluster (*i.e.*, the largest cluster), which is not properly substantiated in the paper. This assumption may be too rigid in some applications. It is possible that the normal data are distributed in two or more clusters that correspond to a few varying normal behaviors. For a simple instance, the network traffic volume is usually high during the daytime and becomes low late in the night. Thus, network traffic volume may display several clusters to represent behaviors exhibiting at different time of the day. In such case, the largest cluster is apparently not where all the normal cases are only residing;

- In this method, one needs to specify the parameter α. The method is rather sensitive to this parameter whose best value is not obvious whatsoever. First, the distance scale between data will be rather different in various subspaces; the distance between any pair of data is naturally increased when it is evaluated in a subspace with higher dimension, compared to in a lower-dimensional subspace. Therefore, specifying an ad-hoc α value for each subspace evaluated is rather tedious and difficult. Second, α is also heavily affected by the number of clusters the clustering method produces, *i.e.*, k. Intuitively, when the number of clusters k is small, D will become relatively large, then α should be set relatively small accordingly, and vice versa.

Recently, an extension of the notion of kNN, called Grid-ODF, from the k nearest objects to the k nearest dense regions is proposed [89]. This method employed the sum of the distances between each data point and its k nearest dense regions to rank data points. This enables the algorithm to measure the outlier-ness of data points from a more global perspective. Grid-ODF takes into account the mechanisms used in detecting both global and local outliers. In the local perspective, human examine the point's immediate neighborhood and consider it as an outlier if its neighborhood density is low. The global observation considers the dense regions where the data points are densely populated in the data space. Specifically, the neighboring density of the point serves as a good indicator of its outlying degree from the local perspective. In the left sub-figure of Figure 2, two square boxes of equal size are used to delimit the neighborhood of points p_1 and p_2. Because the neighboring density of p_1 is less than that of p_2, so the outlying degree of p_1 is larger than p_2. On the other hand, the distance between the point and the dense regions reflects the similarity between this point and the dense regions. Intuitively, the larger such distance is, the more remarkably p is deviated from the main population of the data points and therefore the higher outlying degree it has, otherwise it is not. In the right sub-figure of 2, we can see a dense region and two outlying points, p_1 and p_2. Because the distance between p_1 and the dense region is larger than that between p_2 and the dense region, so the outlying degree of p_1 is larger than p_2.

Based on the above observations, a new measurement of outlying factor of data points, called *Outlying Degree Factor* (ODF), is proposed to measure the outlier-ness of points from both the global and local perspectives. The ODF of a point p is defined as follows:

$$ODF(p) = \frac{k\_DF(p)}{NDF(p)}$$

where $k\_DF(p)$ denotes the average distance between p and its k nearest dense cells and $NDF(p)$ denotes number of points falling into the cell to which p belongs.

In order to implement the computation of ODF of points efficiently, grid structure is used to partition the data space. The main idea of grid-based data space partition is to super-impose a multi-dimensional cube in the data space, with equal-volumed cells. It is characterized by the following advantages. First, $NDF(p)$ can be obtained instantly by simply counting the number of points falling into the cell to which p belongs, without the involvement of any indexing techniques. Secondly, the dense regions can be efficiently identified, thus the computation of $k\_DF(p)$ can be very fast. Finally, based on the density of grid cells, we will be able to select the top n outliers only from a specified number of points viewed as *outlier candidates*, rather than the whole dataset, and the final top n outliers are selected from these outlier candidates based on the ranking of their ODF values.

The number of outlier candidates is typically 9 or 10 times as large as the number of final outliers to be found (*i.e.*, top n) in order to provide a sufficiently large pool for outlier selection. Let us suppose that the size of outlier candidates is $m * n$, where the m is a positive number provided by users. To generate $m * n$ outlier candidates, all the cells containing points are sorted in ascending order based on their densities, and then the points in the first t cells in the sorting list that satisfy the following inequality are selected as the $m * n$ outlier candidates:

$$\sum_{i=1}^{t-1} Den(C_i) \leq m * n \leq \sum_{i=1}^{t} Den(C_i)$$

The kNN-distance methods, which define the top n objects having the highest values of the corresponding outlier-ness metrics as outliers, are advantageous over the local neighborhood methods in that they order the data points based on their relative ranking, rather than on the distance cutoff. Since the value of n, the top outlier users are interested in, can be very small and is relatively independent of the underlying data set, it will be easier for the users to specify compared to the distance threshold λ.

C. Advantages and Disadvantages of Distance-based Methods

The major advantage of distance-based algorithms is that, unlike distribution-based methods, distance-based methods are non-parametric and do not rely on any assumed distribution to fit the data. The distance-based definitions of outliers are fairly straightforward and easy to understand and implement.

Their major drawback is that most of them are not effective in high-dimensional space due to the curse of dimensionality, though one is able to mechanically extend the distance metric, such as Euclidean distance, for high-dimensional data. The high-dimensional data in real applications are very noisy, and the abnormal deviations may be embedded in some lower-dimensional subspaces that cannot be observed in the full data space. Their definitions of a local neighborhood, irrespective of the circular neighborhood or the k nearest neighbors, do not make much sense in high-dimensional space. Since each point tends to be equi-distant with each other as number of dimensions goes up, the degree of outlier-ness of each points are approximately identical and significant phenomenon of deviation or abnormality cannot be observed. Thus, none of the data points can be viewed outliers if the concepts of proximity are used to define outliers. In addition, neighborhood and kNN search in high-dimensional space is a non-trivial and expensive task. Straightforward algorithms, such as those based on nested loops, typically require $O(N^2)$ distance computations. This quadratic scaling means that it will be very difficult to mine outliers as we tackle increasingly larger data sets. This is a major problem for many real databases where there are often millions of records. Thus, these approaches lack a good scalability for large data set. Finally, the existing distance-based methods are not able to deal with data streams due to the difficulty in maintaining a data distribution in the local neighborhood or finding the kNN for the data in the stream.

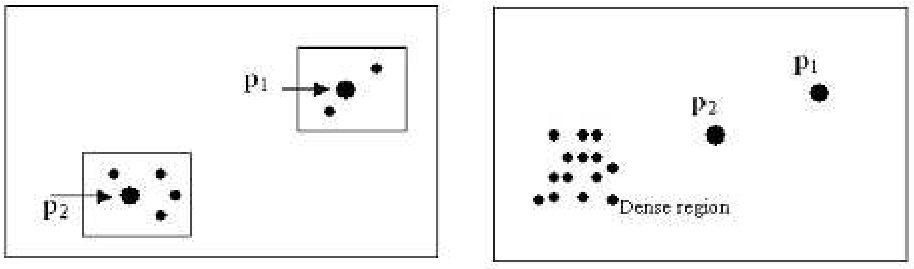


Figure 2. Local and global perspectives of outlier-ness of p_1 and p_2

4.3. Density-based Methods

Density-based methods use more complex mechanisms to model the outlier-ness of data points than distance-based methods. It usually involves investigating not only the local density of the point being studied but also the local densities of its nearest neighbors. Thus, the outlier-ness metric of a data point is relative in the sense that it is normally a ratio of density of this point against the the averaged densities of its nearest neighbors. Density-based methods feature a stronger modeling capability of outliers but require more expensive computation at the same time. What will be discussed in this subsection are the major density-based methods called LOF method, COF method, INFLO method and MDEF method.

A. LOF Method

The first major density-based formulation scheme of outlier has been proposed in [21], which is more robust than the distance-based outlier detection methods. An example is given in [21] (refer to figure 3), showing the advantage of a density-based method over the distance-based methods such as $DB(k, \lambda)$-Outlier. The dataset contains an outlier o, and C_1 and C_2 are two clusters with very different densities. The $DB(k, \lambda)$-Outlier method cannot distinguish o from the rest of the data set no matter what values the parameters k and λ take. This is because the density of o's neighborhood is very much closer to the that of the points in cluster C_1. However, the density-based method, proposed in [21], can handle it successfully.

This density-based formulation quantifies the outlying degree of points using *Local Outlier Factor (LOF)*. Given parameter $MinPts$, LOF of a point p is defined as

$$LOF_{MinPts}(p) = \frac{\sum_{o \in MinPts(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

where $|N_{MinPts}(p)|$ denotes the number of objects falling into the $MinPts$-neighborhood of p and $lrd_{MinPts}(p)$ denotes the *local reachability density* of point p that is defined as the inverse of the average reachability distance based on the $MinPts$ nearest neighbors of p, *i.e.*,

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o \in MinPts(p)} reach\_dist_{MinPts}(p, o)}{|N_{MinPts}(p)|}\right)$$

Further, the reachability distance of point p is defined as

$$reach\_dist_{MinPts}(p, o) = max(MinPts\_distance(o), dist(p, o))$$

Intuitively speaking, LOF of an object reflects the density contrast between its density and those of its neighborhood. The neighborhood is defined by the distance to the $MinPts^{th}$ nearest neighbor. The local outlier factor is a mean value of the ratio of the density distribution estimate in the neighborhood of the object analyzed to the distribution densities of its neighbors [21]. The lower the density of p and/or the higher the densities of p's neighbors, the larger the value of $LOF(p)$, which indicates that p has a higher degree of being an outlier. A similar outlier-ness metric to LOF, called OPTICS-OF, was proposed in [20].

Unfortunately, the LOF method requires the computation of LOF for all objects in the data set which is rather expensive because it requires a large number of kNN search. The high cost of computing LOF for each data point p is caused by two factors. First, we have to find the $MinPts^{th}$ nearest neighbor of p in order to specify its neighborhood. This resembles to computing D^k in detecting D^k_n-Outliers. Secondly, after the $MinPts^{th}$-neighborhood of p has been determined, we have to further find the $MinPts^{th}$-neighborhood for each data points falling into the $MinPts^{th}$-neighborhood of p. This amounts to $MinPts^{th}$ times in terms of computation efforts as computing D^k when we are detecting D^k_n-Outliers.

It is desired to constrain a search to only the top n outliers instead of computing the LOF of every object in the database. The efficiency of this algorithm is boosted by an efficient micro-cluster-based local outlier mining algorithm proposed in [52].

LOF ranks points by only considering the neighborhood density of the points, thus it may miss out the

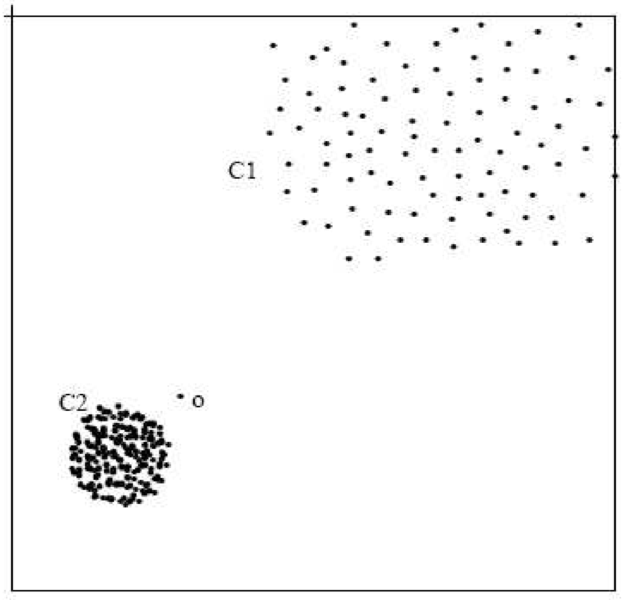


Figure 3. A sample dataset showing the advantage of LOF over $DB(k, \lambda)$-Outlier

potential outliers whose densities are close to those of their neighbors. Furthermore, the effectiveness of this algorithm using LOF is rather sensitive to the choice of $MinPts$, the parameter used to specify the local neighborhood.

B. COF Method

As LOF method suffers the drawback that it may miss those potential outliers whose local neighborhood density is very close to that of its neighbors. To address this problem, Tang *et al.* proposed a new *Connectivity-based Outlier Factor* (COF) scheme that improves the effectiveness of LOF scheme when a pattern itself has similar neighborhood density as an outlier [87]. In order to model the connectivity of a data point with respect to a group of its neighbors, a set-based nearest path (SBN-path) and further a set-based nearest trail (SBN-trail), originated from this data point, are defined. This SNB trail stating from a point is considered to be the pattern presented by the neighbors of this point. Based on SNB trail, the cost of this trail, a weighted sum of the cost of all its constituting edges, is computed. The final outlier-ness metric, COF, of a point p with respect to its k-neighborhood is defined as

$$COF_k(p) = \frac{|N_k(p)| * ac\_dist_{N_k(p)}(p)}{\sum_{o \in N_k(p)} ac\_dist_{N_k(o)}(o)}$$

where $ac\_dist_{N_k(p)}(p)$ is the average chaining distance from point p to the rest of its k nearest neighbors, which is the weighted sum of the cost of the SBN-trail starting from p.

It has been shown in [87] that COF method is able to detect outlier more effectively than LOF method for some cases. However, COF method requires more expensive computations than LOF and the time complexity is in the order of $O(N^2)$ for high-dimensional datasets.

C. INFLO Method

Even though LOF is able to accurately estimate outlier-ness of data points in most cases, it fails to do so in some complicated situations. For instance, when outliers are in the location where the density distributions in the neighborhood are significantly different, this may result in a wrong estimation. An example where LOF fails to have an accurate outlier-ness estimation for data points has been given in [53]. The example is presented in Figure 4. In this example, data p is in fact part of a sparse cluster $C2$ which is near the dense cluster $C1$. Compared to objects q and r, p obviously displays less outlier-ness. However, if LOF is used in this case, p could be mistakenly regarded to having stronger outlier-ness than q and r.

Authors in [53] pointed out that this problem of LOF is due to the inaccurate specification of the space where LOF is applied. To solve this problem of LOF, an improved method, called INFLO, is proposed [53]. The idea of INFLO is that both the nearest neighbors (NNs) and reverse nearest neighbors (RNNs) of a data point are taken into account in order to get a better estimation of the neighborhood's density distribution. The RNNs of an object p are those data points that have p as one of their k nearest neighbors. By considering the symmetric neighborhood relationship of both NN and RNN, the space of an object influenced by other objects is well determined. This space is called the *k-influence space* of a data point. The outlier-ness of a data point, called INFLuenced Outlierness (INFLO), is quantified. INFLO of a data point p is defined as

$$INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)}$$

INFLO is by nature very similar to LOF. With respect to a data point p, they are both defined as the ratio of p's its density and the average density of its neighboring objects. However, INFLO uses only the data points in its k-influence space for calculating the density ratio. Using INFLO, the densities of its neighborhood will be reasonably estimated, and thus the outliers found will be more meaningful.

D. MDEF Method

In [77], a new density-based outlier definition, called Multi-granularity Deviation Factor (MEDF), is proposed. Intuitively, the MDEF at radius r for a point p_i is the relative deviation of its local neighborhood density from the average local neighborhood density in its r-neighborhood. Let $n(p_i, \alpha r)$ be the number of objects in the αr-neighborhood of p_i and $\hat{n}(p_i, r, \alpha)$ be the average, over all objects p in the r-neighborhood of p_i, of $n(p, \alpha r)$. In the example given by Figure 5, we have $n(p_i, \alpha r) = 1$, and $\hat{n}(p_i, r, \alpha) = (1 + 6 + 5 + 1)/4 = 3.25$.

MDEF of p_i, given r and α, is defined as

$$MDEF(p_i, r, \alpha) = 1 - \frac{n(p_i, \alpha r)}{\hat{n}(pi, r, \alpha)}$$

where $\alpha = \frac{1}{2}$. A number of different values are set for the sampling radius r and the minimum and the maximum values for r are denoted by r_{min} and r_{max}. A point is flagged as an outliers if for any $r \in [r_{min}, r_{max}]$, its MDEF is sufficient large.

E. Advantages and Disadvantages of Density-based Methods

The density-based outlier detection methods are generally more effective than the distance-based methods. However, in order to achieve the improved effectiveness, the density-based methods are more complicated and computationally expensive. For a data object, they have to not only explore its local density but also that of its neighbors. Expensive kNN search is expected for all the existing methods in this category. Due to the inherent complexity and non-updatability of their outlier-ness measurements used, LOF, COF, INFLO and MDEF cannot handle data streams efficiently.

4.4. Clustering-based Methods

The final category of outlier detection algorithm for relatively low dimensional static data is clustering-based. Many data-mining algorithms in literature find outliers as a by-product of clustering algorithms [6, 11, 13, 46, 101] themselves and define outliers as points that do not lie in or located far apart from any clusters. Thus, the clustering techniques implicitly define outliers as the background noise of clusters. So far, there are numerous studies on clustering, and some of them are equipped with some mechanisms to reduce the adverse effect of outliers, such as CLARANS [73], DBSCAN [37], BIRCH [101], WaveCluster [82]. More recently, we have seen quite a few clustering techniques tailored towards subspace clustering for high-dimensional data including CLIQUE [6] and HPStream [9].

Next, we will review several major categories of clustering methods, together with the analysis on their advantages and disadvantages and their applicability in dealing with outlier detection problem for high-dimensional data streams.

A. Partitioning Clustering Methods

The partitioning clustering methods perform clustering by partitioning the data set into a specific number of clusters. The number of clusters to be obtained, denoted by k, is specified by human users. They typically start with an initial partition of the dataset and then iteratively optimize the objective function until it reaches the optimal for the dataset. In the clustering process, center of the clusters (centroid-based methods) or the point which is located nearest to the cluster center (medoid-based methods) is used to represent a cluster. The representative partitioning clustering methods are PAM, CLARA, k-means and CLARANS.

PAM [62] uses a k-medoid method to identify the clusters. PAM selects k objects arbitrarily as medoids and swap with objects until all k objects qualify as medoids. PAM compares an object with entire dataset to find a medoid, thus it has a slow processing time with a complexity of $\mathcal{O}(k(N-k)^2)$, where N is number of data in the data set and k is the number of clusters.

CLARA [62] tries to improve the efficiency of PAM. It draws a sample from the dataset and applies PAM on the sample that is much smaller in size than the the whole dataset.

k-means [68] initially choose k data objects as seeds from the dataset. They can be chosen randomly or in a way such that the points are mutually farthest apart. Then, it examines each point in the dataset and assigns it to one of the clusters depending on the minimum distance. The centroid's position is recalculated and updated the moment a point is added to the cluster and this continues until all the points are grouped into the final clusters. The k-means algorithm is relatively

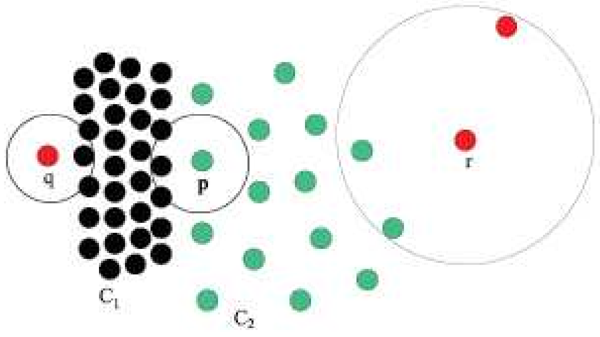


Figure 4. An example where LOF does not work

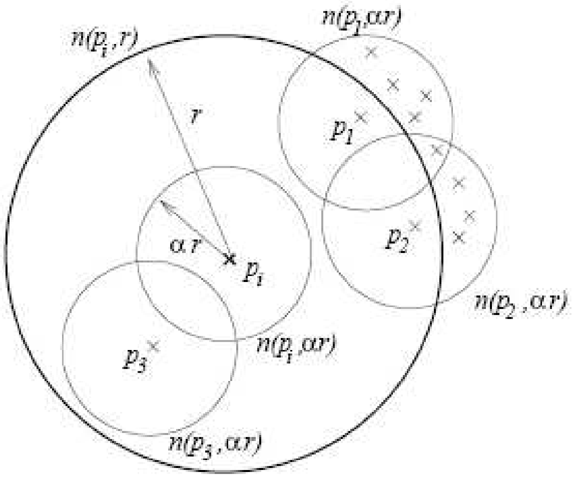


Figure 5. Definition of MDEF

scalable and efficient in processing large datasets because the computational complexity is $\mathcal{O}(nkt)$, where n is total number of points, k is the number of clusters and t is the number of iterations of clustering. However, because it uses a centroid to represent each cluster, k-means suffers the inability to correctly cluster with a large variation of size and arbitrary shapes, and it is also very sensitive to the noise and outliers of the dataset since a small number of such data will substantially effect the computation of mean value the moment a new object is clustered.

CLARANS [73] is an improved k-medoid method, which is based on randomized search. It begins with a random selection of k nodes, and in each of following steps, compares each node to a specific number of its neighbors in order to find a local minimum. When one local minimum is found, CLARANS continues to repeat this process for another minimum until a specific number of minima have been found. CLARANS has been experimentally shown to be more effective than both PAM and CLEAR. However, the computational complexity of CLARANS is close to quadratic w.r.t the number of points [88], and it is prohibitive for clustering large database. Furthermore, the quality of clustering result is dependent on the sampling method, and it is not stable and unique due to the characteristics of randomized search.

B. Hierarchical Clustering Methods

Hierarchical clustering methods essentially constructs a hierarchical decomposition of the whole dataset. It can be further divided into two categories based on how this dendrogram is operated to generate clusters, *i.e.*, *agglomerative methods* and *divisive methods*. An agglomerative method begins with each point as a distinct cluster and merges two closest clusters in each subsequent step until a stopping criterion is met. A divisive method, contrary to an agglomerative method, begins with all the point as a single cluster and splits it in each subsequent step until a stopping criterion is met. Agglomerative methods are seen more popular in practice. The representatives of hierarchical methods are MST clustering, CURE and CHAMELEON.

MST clustering [92] is a graph-based divisive clustering algorithm. Given n points, a MST is a set of edges that connects all the points and has a minimum total length. Deletion of edges with larger lengths will subsequently generate a specific number of clusters. The overhead for MST clustering is determined by the Euclidean MST construction, which is $\mathcal{O}$(\↕≀}\) in time complexity, thus MST algorithm can be used for scalable clustering. However, MST algorithm can only work well on the clean dataset and are sensitive to outliers. The intervention of outliers, termed "chaining-effect" (that is, a line of outliers between two distinct clusters will make these two clusters be marked as one cluster due to its adverse effect), will seriously degrade the quality of the clustering results.

CURE [46] employs a novel hierarchical clustering algorithm in which each cluster is represented by a constant number of well-distributed points. A random sample drawn from the original dataset is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters. The multiple representative points for each cluster are picked to be as disperse as possible and shrink towards the center using a pre-specified shrinking factor. At each step of the algorithm, the two clusters with the closest pair of representative (this pair of representative points are from different clusters) points are merged. Usage of multiple points representing a cluster enables CURE to well capture the shape of clusters and makes it suitable for clusters with non-spherical shapes and wide variance in size. The shrinking factor helps to dampen the adverse effect of outliers. Thus, CURE is more robust to outliers and identifies clusters having arbitrary shapes.

CHAMELEON [55] is a clustering technique trying to overcome the limitation of existing agglomerative hierarchical clustering algorithms that the clustering is irreversible. It operates on a sparse graph in which nodes represent data points and weighted edges represent similarities of among the data points. CHAMELEON first uses a graph partition algorithm to cluster the data points into a large number of relatively small sub-clusters. It then employs an agglomerative hierarchical clustering algorithm to genuine clusters by progressively merging these sub-clusters. The key feature of CHAMELEON lies in its mechanism determining the similarity between two sub-clusters in sub-cluster merging. Its hierarchical algorithm takes into consideration of both inter-connectivity and closeness of clusters. Therefore, CHAMELEON can dynamically adapt to the internal characteristics of the clusters being merged.

C. Density-based Clustering Methods

The density-based clustering algorithms consider normal clusters as dense regions of objects in the data space that are separated by regions of low density. Human normally identify a cluster because there is a relatively denser region compared to its sparse neighborhood. The representative density-based clustering algorithms are DBSCAN and DENCLUE.

The key idea of DBSCAN [37] is that for each point in a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN introduces the notion of "density-reachable points" and based on which performs clustering. In DBSCAN, a cluster is a maximum set of density-reachable points w.r.t. parameters Eps and $MinPts$, where Eps is the given radius and $MinPts$ is the minimum number of points required to be in the Eps-neighborhood. Specifically, to discover clusters in the dataset, DBSCAN examines the Eps-neighborhood of each point in the dataset. If the Eps-neighborhood of a point p contains more than $MinPts$, a new cluster with p as the core object is generated. All the objects from within this Eps-neighborhood are then assigned to this cluster. All this newly entry points will also go through the same process to gradually grow this cluster. When there is no more core object can be found, another core object will be initiated and another cluster will grow. The whole clustering process terminates when there are no new points can be added to any clusters. As the clusters discovered are dependent on the specification of the parameters, DBSCAN relies on the user's ability to select a good set of parameters. DBSCAN outperforms CLARANS by a factor of more than 100 in terms of efficiency [37]. DBSCAN is also powerful in discovering of clusters with arbitrary shapes. The drawbacks DBSCAN suffers are: (1) It is subject to adverse effect resulting from "chaining-effect"; (2) The two parameters used in DBSCAN, *i.e.*, Eps and $MinPts$, cannot be easily decided in advance and require a tedious process of parameter tuning.

DENCLUE [51] performs clustering based on density distribution functions, a set of mathematical functions used to model the influence of each point within its neighborhood. The overall density of the data space can be modeled as sum of influence function of all data points. The clusters can be determined by density attractors. Density attractors are the local maximum of the overall density function. DENCLUE has advantages that it can well deal with dataset with a large number of noises and it allows a compact description of clusters of arbitrary shape in high-dimensional datasets. To facilitate the computation of the density function, DENCLUE makes use of grid-like structure. Noted that even though it uses grids in clustering, DENCLUE is fundamentally different from grid-based clustering algorithm in that grid-based clustering algorithm uses grid for summarizing information about the data points in each grid cell, while DENCLUE uses such structure to effectively compute the sum of influence functions at each data point.

D. Grid-based Clustering Methods

Grid-based clustering methods perform clustering based on a grid-like data structure with the aim of enhancing the efficiency of clustering. It quantizes the space into a finite number of cells which form a grid structure on which all the operations for clustering are performed. The main advantage of the approaches in this category is their fast processing time which is typically only dependent on the number of cells in the quantized space, rather than the number of data objects. The representatives of grid-based clustering algorithms are STING, WaveCluster and DClust.

STING [88] divides the spatial area into rectangular grids, and builds a hierarchical rectangle grids structure. It scans the dataset and computes the necessary statistical information, such as mean, variance, minimum, maximum, and type of distribution, of each grid. The hierarchical grid structure can represent the statistical information with different resolutions at different levels. The statistical information in this hierarchical structure can be used to answer queries. The likelihood that a cell is relevant to the query at some confidence level is computed using the parameters of this cell. The likelihood can be defined as the proportion of objects in this cell that satisfy the query condition. After the confidence interval is obtained, the cells are labeled as relevant or irrelevant based on some confidence threshold. After examining the current layer, the clustering proceeds to the next layer and repeats the process. The algorithm will subsequently only examine the relevant cells instead of all the cells. This process terminates when all the layers have been examined. In this way, all the relevant regions (clusters) in terms of query are found and returned.

WaveCluster [82] is grid-based clustering algorithm based on wavelet transformation, a commonly used technique in signal processing. It transforms the multi-dimensional spatial data to the multi-dimensional signal, and it is able to identify dense regions in the transformed domain that are clusters to be found.

In DClust [95], the data space involved is partitioned into cells with equal size and data points are mapped into the grid structure. A number of representative points of the database are picked using the density criterion. A *Minimum Spanning Tree* (MST) of these representative points, denoted as R-MST, is built. After the R-MST has been constructed, multi-resolution clustering can be easily achieved. Suppose a user wants to find k clusters. A graph search through the R-MST is initiated, starting from the largest cost edge, to the lowest cost edge. As an edge is traversed, it is marked as deleted from the R-MST. The number of partitions resulting from the deletion is computed. The process stops when the number of partitions reaches k. Any change in the value of k simply implies re-initiating the search-and-marked procedure on the R-MST. Once the R-MST has been divided into k partitions, we can now propagate this information to the original dataset so that each point in the dataset is assigned to one and only one partition/cluster. DClust is equipped with more robust outlier elimination mechanisms to identify and filter the outliers during the various stages of the clustering process. First, DClust uses a uniform random sampling approach to sample the large database. This is effective in ruling out the majority of outliers in the database. Hence, the sample database obtained will be reasonably clean; Second, DClust employs a grid structure to identify representative points. Grid cells whose density is less than the threshold are pruned. This pre-filtering step ensures that the R-MST constructed is an accurate reflection of the underlying cluster structure. Third, the clustering of representative points may cause a number of the outliers that are in close vicinity to form a cluster. The number of points in such outlier clusters will be much smaller than the number of points in the normal clusters. Thus, any small clusters of representative points will be treated as outlier clusters and eliminated. Finally, when the points in the dataset are labeled, some of these points may be quite far from any representative point. DClust will regard such points as outliers and filter them out in the final clustering results.

E. Advantages and Disadvantage of Clustering-based Methods

Detecting outliers by means of clustering analysis is quite intuitive and consistent with human perception of outliers. In addition, clustering is a well-established research area and there have been abundant clustering algorithms that users can choose from for performing clustering and then detecting outliers.

Nevertheless, many researchers argue that, strictly speaking, clustering algorithms should not be considered as outlier detection methods, because their objective is only to group the objects in dataset such that clustering functions can be optimized. The aim to eliminate outliers in dataset using clustering is only to dampen their adverse effect on the final clustering result. This is in contrast to the various definitions of outliers in outlier detection which are more objective and independent of how clusters in the input data set are identified. One of the major philosophies in designing new outlier detection approaches is to directly model outliers and detect them without going though clustering the data first. In addition, the notions of outliers in the context of clustering are essentially binary in nature, without any quantitative indication as to how outlying each object is. It is desired in many applications that the outlier-ness of the outliers can be quantified and ranked.

5. Outlier Detection Methods for High Dimensional Data

There are many applications in high-dimensional domains in which the data can contain dozens or even hundreds of dimensions. The outlier detection techniques we have reviewed in the preceding sections use various concepts of proximity in order to find the outliers based on their relationship to the other points in the data set. However, in high-dimensional space, the data are sparse and concepts using the notion of proximity fail to achieve most of their effectiveness. This is due to the curse of dimensionality that renders the high-dimensional data tend to be equi-distant to each other as dimensionality increases. They does not consider the outliers embedded in subspaces and are not equipped with the mechanism for detecting them.

5.1. Methods for Detecting Outliers in High-dimensional Data

To address the challenge associated with high data dimensionality, two major categories of research work have been conducted. The first category of methods project the high dimensional data to lower dimensional data. Dimensionality deduction techniques, such as *Principal Component Analysis*(PCA), *Independent Component Analysis* (ICA), *Singular Value Decomposition* (SVD), etc can be applied to the high-dimensional data before outlier detection is performed. Essentially, this category of methods perform feature selection and can be considered as the pre-processing work for outlier detection. The second category of approaches is more promising yet challenging. They try to re-design the mechanism to accurately capture the proximity relationship between data points in the high-dimensional space [14].

A. Sparse Cube Method. Aggarwal *et al.* conducted some pioneering work in high-dimensional outlier detection [15][14]. They proposed a new technique for outlier detection that finds outliers by observing the density distributions of projections from the data. This new definition considers a point to be an outlier if in some lower-dimensional projection it is located in a local region of abnormally low density. Therefore, the outliers in these lower-dimensional projections are detected by simply searching for these projections featuring lower density. To measure the sparsity of a lower-dimensional projection quantitatively, the authors proposed the so-called *Sparsity Coefficient*. The computation of Sparsity Coefficient involves a grid discretization of the data space and making an assumption of normal distribution for the data in each cell of the hypercube. Each attribute of the data is divided into φ equi-depth ranges. In each range, there is a fraction $f = 1/\varphi$ of the data. Then, a k-dimensional cube is made of ranges from k different dimensions. Let N be the dataset size and $n(D)$ denote the number of objects in a k-dimensional cube D. Under the condition that attributes were statistically independent, the Sparsity Coefficient $S(D)$ of the cube D is defined as:

$$S(D) = \frac{n(D) - N * f^k}{\sqrt{N * f^k * (1 - f^k)}}$$

Since there are no closure properties for Sparsity Coefficient, thus no fast subspace pruning can be performed and the lower-dimensional projection search problem becomes a NP-hard problem. Therefore, the authors employ evolutionary algorithm in order to solve this problem efficiently. After lower-dimensional projections have been found, a post-processing phase is required to map these projections into the data points; all the sets of data points that contain in the abnormal projections reported by the algorithm.

B. Example-based Method. Recently, an approach using outlier examples provided by users are used to detect outliers in high-dimensional space [96][97]. It adopts an *"outlier examples → subspaces → outliers"* manner to detect outliers. Specifically, human users or domain experts first provide the systems with a few initial outlier examples. The algorithm finds the subspaces in which most of these outlier examples exhibit significant outlier-ness. Finally, other outliers are detected from these subspaces obtained in the previous step. This approach partitions the data space into equi-depth cells and employs the Sparsity Coefficient proposed in [14] to measure the outlier-ness of outlier examples in each subspace of the lattice. Since it is untenable to exhaustively search the space lattice, the author also proposed to use evolutionary algorithms for subspace search. The fitness of a subspace is the average Sparsity Coefficients of all cubes in that subspace to which the outlier examples belong. All the objects contained in the cubes which are sparser than or as sparse as cubes containing outlier examples in the subspace are detected as outliers.

However, this method is limited in that it is only able to find the outliers in the subspaces where most of the given user examples are outlying significantly. It cannot detect those outliers that are embedded in other subspaces. Its capability for effective outlier detection is largely depended on the number of given examples and, more importantly, how these given examples are similar to the majority of outliers in the dataset. Ideally, this set of user examples should be a good sample of all the outliers in the dataset. This method works poorly when the number of user examples is quite small and cannot provide enough clues as to where the majority of outliers in the dataset are. Providing such a good set of outlier examples is a difficult task whatsoever. The reasons are two-fold. First, it is not trivial to obtain a set of outlier examples for a high-dimensional data set. Due to a lack of visualization aid in high-dimensional data space, it is not obvious at all to find the initial outlier examples unless they are detected by some other techniques. Secondly and more importantly, even when a set of outliers have already been obtained, testing the representativeness of this outlier set is almost impossible. Given these two strong constraints, this approach becomes inadequate in detecting outliers in high-dimensional datasets. It will miss out those projected outliers that are not similar to those given outlier examples.

C. Outlier Detection in Subspaces. Since outlier-ness of data points mainly appear significant in some subspaces of moderate dimensionality in high-dimensional space and the quality of the outliers detected varies in different subspaces consisting of different combinations of dimension subsets. The authors in [24] employ evolutionary algorithm for feature selection (find optimal dimension subsets which represent the original dataset without losing information for unsupervised learning task of outlier detection as well as clustering). This approach is a wrapper algorithm in which the dimension subsets are selected such that the quality of outlier detected or the clusters generated can be optimized. The originality of this work is to combine the evolutionary algorithm with the data visualization technique utilizing parallel coordinates to present evolution results interactively and allow users to actively participate in evolutionary algorithm searching to achieve a fast convergence of the algorithm.

D. Subspace Outlier Detection for Categorical Data. Das *et al.* study the problem of detecting anomalous records in categorical data sets [33]. They draw on a probability approach for outlier detection. For each record in the data set, the probabilities for the occurrence of different subsets of attributes are investigated. A data record is labeled as an outlier if the occurrence probability for the values of some of its attribute subsets is quite low. Specifically, the probability for two subsets of attributes a_t and b_t to occur together in a record, denoted by $r(a_t, b_t)$, is quantified as:

$$r(a_t, b_t) = \frac{P(a_t, b_t)}{P(a_t)P(b_t)}$$

Due to the extremely large number of possible attribute subsets, only the attribute subsets with a length not exceeding than k are studied.

Because it always evaluates pairs of attribute subsets, each of which contain at least one attribute, therefore, this method will miss out the abnormality evaluation for 1-dimensional attribute subsets. In addition, due to the exponential growth of the number of attribute subsets w.r.t k, the value of k is set typically small in this method. Hence, this method can only cover attribute subsets not larger than $2k$ for a record (this method evaluates a pair of attribute subsets at a time). This limits the ability of this method for detecting records that have outlying attribute subsets larger than $2k$.

5.2. Outlying Subspace Detection for High-dimensional Data

All the outlier detection algorithms that we have discussed so far, regardless of in low or high dimensional scenario, invariably fall into the framework of detecting outliers in a specific data space, either in full space or subspace. We term these methods as *"space → outliers"* techniques. For instance, outliers are detected by first finding locally sparse subspaces [14], and the so-called Strongest/Weak Outliers are discovered by first finding the Strongest Outlying Spaces [59].

A new research problem called *outlying subspace detection* for multiple or high dimensional data has been identified recently in [94][99][93]. The major task of outlying subspace detection is to find those subspaces (subset of features) in which the data points of interest exhibit significant deviation from the rest of population. This problem can be formulated as follows: given a data point or object, find the subspaces in which this data is considerably dissimilar, exceptional or inconsistent with respect to the remaining points or objects. These points under study are called *query points*, which are usually the data that users are interested in or concerned with. As in [94][99], a distance threshold T is utilized to decide whether or not a data point deviates significantly from its neighboring points. A subspace s is called an outlying subspace of data point p if $OD_s(p) \geq T$, where OD is the outlier-ness measurement of p.

Finding the correct subspaces so that outliers can be detected is informative and useful in many practical applications. For example, in the case of designing a training program for an athlete, it is critical to identify the specific subspace(s) in which an athlete deviates from his or her teammates in the daily training performances. Knowing the specific weakness (subspace) allows a more targeted training program to be designed. In a medical system, it is useful for the Doctors to identify from voluminous medical data the subspaces in which a particular patient is found abnormal and therefore a corresponding medical treatment can be provided in a timely manner.

The unique feature of the problem of outlying subspace detection is that, instead of detecting outliers in specific subspaces as did in the classical outlier detection techniques, it involves searching from the space lattice for the associated subspaces whereby the given data points exhibit abnormal deviations. Therefore, the problem of outlying subspace detection is called an *"outlier → spaces"* problem so as to distinguish the classical outlier detection problem which is labeled as a *"space → outliers"* problem. It has been theoretically and experimentally shown that the conventional outlier detection methods, irrespectively dealing with low or high-dimensional data, cannot successfully cope with the problem of outlying subspace detection problem in [94]. The existing high-dimensional outlier detection techniques, *i.e.*, find outliers in given subspaces, are theoretically applicable to solve the outlying detection problem. To do this, we have to detect outliers in all subspaces and a search in all these subspaces is needed to find the set of outlying subspaces of p, which are those subspaces in which p is in their respective set of outliers. Obviously, the computational and space costs are both in an exponential order of d, where d is the number of dimensions of the data point. Such an exhaustive space searching is rather expensive in high-dimensional scenario. In addition, they usually only return the top n outliers in a given subspace, thus it is impossible to check whether or not p is an outlier in this subspace if p is not in this top n list. This analysis provides an insight into the inherent difficulty of using the existing high-dimensional outlier detection techniques to solve the new outlying subspace detection problem.

A. HighDoD

Zhang *et al.* proposed a novel dynamic subspace search algorithm, called HighDoD, to efficiently identify the outlying subspaces for the given query data points [94][99]. The outlying measure, *OD*, is based on the sum of distances between a data and its k nearest neighbors [10]. This measure is simple and independent of any underlying statistical and distribution characteristics of the data points. The following two heuristic pruning strategies employing upward-and downward closure property are proposed to aid in the search for outlying subspaces: *If a point p is not an outlier in a subspace s, then it cannot be an outlier in any subspace that is a subset of s. If a point p is an outlier in a subspace s, then it will be an outlier in any subspace that is a superset of s*. These two properties can be used to quickly detect the subspaces in which the point is not an outlier or the subspaces in which the point is an outlier. All these subspaces can be removed from further consideration in the later stage of the search process. A fast dynamic subspace search algorithm with a sample-based learning process is proposed. The learning process aims to quantitize the prior probabilities for upward- and downward pruning in each layer of space lattice. The *Total Saving Factor* (TSF) of each layer of subspaces in the lattice, used to measure the potential advantage in saving computation, is dynamically updated and the search is performed in the layer of lattice that has the highest TSF value in each step of the algorithm.

However, HighDoD suffers the following major limitations. First, HighDoD relies heavily on the closure (monotonicity) property of the outlying measurement of data points, termed OD, to perform the fast bottom-up or top-down subspace pruning in the space lattice, which is the key technique HighDoD utilizes for speeding up subspace search. Under the definition of OD, a subspace will always be more likely to be an outlying subspace than its subset subspaces. This is because that OD of data points will be naturally increased when the dimensionality of the subspaces under study goes up. Nevertheless, this may not be a very accurate measurement. The definition of a data point's outlier-ness makes more sense if its measurement can be related to other points, meaning that the averaged level of the measurement for other points in the same subspace should be taken into account simultaneously in order to make the measurement statistically significant. Therefore, the design of a new search method is desired in this situation. Secondly, HighDoD labels each subspace in a binary manner, either an outlying subspace or a non-outlying one, and most subspaces are pruned away before their outlying measurements are virtually evaluated in HighDoD. Thus, it is not possible for HighDoD to return a ranked list of the detected outlying subspaces. Apparently, a ranked list will be more informative and useful than an unranked one in many cases. Finally, a human-user defined cutoff for deciding whether a subspace is outlying or not with respect to a query point is used. This parameter will define the "outlying front" (the boundary between the outlying subspaces and the non-outlying ones). Unfortunately, the value of this parameter cannot be easily specified due to the lack of prior knowledge concerning the underlying distribution of data point that maybe very complex in the high-dimensional spaces.

B. SOF Method

In [100], a novel technique based on genetic algorithm is proposed to solve the outlying subspace detection problem and well copes with the drawbacks of the existing methods. A new metric, called *Subspace Outlying Factor (SOF)*, is developed for measuring the outlying degree of each data point in different subspaces. Based on SOF, a new definition of outlying subspace, called SOF Outlying Subspaces, is proposed. Given an input dataset D, parameters n and k, a subspace s is a SOF Outlying Subspace for a given query data point p if there are no more than $n-1$ other subspaces s' such that $SOF(s', p) > SOF(s, p)$. The above definition is equivalent to say that the top n subspaces having the largest SOF values are considered to be outlying subspaces. The parameters used in defining SOF Outlying Subspaces are easy to be specified, and do not require any prior knowledge about the data distribution of the dataset. A genetic algorithm (GA) based method is proposed for outlying subspace detection. The upward and downward closure property is no longer required in the GA-based method, and the detected outlying subspaces can be ranked based on their fitness function values. The concepts of the lower and upper bounds of D^k, the distance between a given point and its k^{th} nearest neighbor, are proposed. These bounds are used for a significant performance boost in the method by providing a quick approximation of the fitness of subspaces in the GA. A technique is also proposed to compute these bounds efficiently using the so-called kNN Look-up Table.

5.3. Clustering Algorithms for High-dimensional Data

We have witnessed some recent developments of clustering algorithms towards high-dimensional data. As clustering provides a possible, even though not the best, means to detect outliers, it is necessary for us to review these new developments. The representative methods for clustering high-dimensional data are CLIQUE and HPStream.

A. CLIQUE

CLIQUE [7] is a grid-based clustering method that discretizes the data space into non-overlapping rectangular units, which are obtained by partitioning every dimension into a specific number of intervals of equal length. A unit is dense if the fraction of total data points contained in this unit is greater than a threshold. Clusters are defined as unions of connected dense units within a subspace. CLIQUE first identifies a subspace that contains clusters. A bottom-up algorithm is used that exploits the monotonicity of the clustering criterion with respect to dimensionality: if a k-dimensional unit is dense, then so are its projections in (k-1) -dimensional space. A candidate generation procedure iteratively determines the candidate k-dimensional units C_k after determining the (k-1)-dimensional dense units D_{k-1}. A pass is made over the data to determine those candidates units that are dense D_k. A depth-first search algorithm is then used to identify clusters in the subspace: it starts with some unit u in D, assign it the first cluster label number, and find all the units it is connected to. Then, if there are still units in D that have yet been visited, it finds one and repeats the procedure. CLIQUE is able to automatically finds dense clusters in subspaces of high-dimensional dataset. It can produce identical results irrespective of the order in which input data are presented and not presume any specific mathematical form of data distribution. However, the accuracy of this clustering method maybe degraded due to the simplicity of this method. The clusters obtained are all of the rectangular shapes, which is obviously not consistent with the shape of natural clusters. In addition, the subspaces obtained are dependent on the choice of the density threshold. CLIQUE uses a global density threshold (*i.e.*, a parameter that is used for all the subspaces), thus it is difficult to specify its value especially in high-dimensional subspaces due to curse of dimensionality. Finally, the subspaces obtained are those where dense units exist, but this has nothing to do with the existence of outliers. As a result, CLIQUE is not suitable for detecting projected outliers.

B. HPStream

In order to find the clusters embedded in the subspaces of high-dimensional data space in data streams, a new clustering method, called HPStream, is proposed [9]. HPStream introduces the concept of *projected clustering* to data streams as significant and high-quality clusters only exist in some low-dimensional subspaces. The basic idea of HPStream is that it does not only find clusters but also updates the set of dimensions associated with each cluster where more compact clusters can be found. The total number of clusters obtained in HPStream is initially obtained through $k-$means clustering and the initial set of dimensions associated with each of these k clusters is the full set of dimensions of the data stream. As more streaming data arrive, the set of dimensions for each cluster evolves such that each cluster can become more compact with a smaller radius.

HPStream is innovative in finding clusters that are embedded in subspaces for high-dimensional data streams. However, the number of subspaces returned by HPStream is equal to the number of clusters obtained that is typically of a small value. Consequently, if HPStream is applied to detect projected outliers, then it will only be able to detect the outliers in those subspaces returned and miss out a significant potions of outliers existing in other subspaces that are not returned by HPStream. Of course, it is possible to increase the number of subspaces returned in order to improve the detection rate. However, the increase of subspaces will imply an increase of the number of clusters accordingly. An unreasonably large number of clusters is not consistent with the formation of natural clusters and will therefore affect the detection accuracy of projected outliers.

6. Outlier Detection Methods for Data Streams

The final major category of outlier detection methods we will discuss in this section are those outlier detection methods for handling data streams. We will first discuss Incremental LOF, and then the outlier detection methods for sensor networks that use Kernel density function. The incremental clustering methods that can handle continuously arriving data will also be covered at the end of this subsection.

A. Incremental LOF Method

Since LOF method is not able to handle data streams, thus an incremental LOF algorithm, appropriate for detecting outliers from dynamic databases where frequently data insertions and deletions occur, is proposed in [78]. The proposed incremental LOF algorithm provides an equivalent detection performance as the iterated static LOF algorithm (applied after insertion of each data record), while requiring significantly less computational time. In addition, the incremental LOF algorithm also dynamically updates the profiles of data points. This is an appealing property, since data profiles may change over time. It is shown that insertion of new data points as well as deletion of obsolete points influence only limited number of their nearest neighbors and thus insertion/deletion time complexity per data point does not depend on the total number of points N[78].

The advantage of Incremental LOF is that it can deal with data insertions and deletions efficiently. Nevertheless, Incremental LOF is not economic in space. The space complexity of this method is in the order of the data that have been inserted but have not been deleted. In other words, Incremental LOF has to maintain the whole length of data stream in order to deal with continuously arriving data because it does not utilize any compact data summary or synopsis. This is clearly not desired for data stream applications that are typically subject to explicit space constraint.

B. Outlier Detection Methods for Sensor Networks

There are a few recent anomaly detection methods for data streams. They mainly come from sensor networks domain such as [80] and [25]. However, the major effort taken in these works is the development of distributable outlier detection methods from distributed data streams and does not deal with the problem of outlier detection in subspaces of high-dimensional data space. Palpanas *et al.* proposed one of the first outlier detection methods for distributed data streams in the context of sensor networks [80]. The author classified the sensor nodes in the network as the low capacity and high capacity nodes, through which a multi-resolution structure of the sensor network is created. The high capacity nodes are nodes equipped with relatively strong computational strength that can detect local outliers. The Kernel density function is employed to model local data distribution in a single or multiple dimensions of space. A point is detected as an outlier if the number of values that have fallen into its neighborhood (delimited by a sphere of radius r) is less than an application-specific threshold. The number of values in the neighborhood can be computed by the Kernel density function. Similarly, the authors in [25] also emphasize the design of distributed outlier detection methods. Nevertheless, this work employs a number of different commonly used outlier-ness metric such as the distance to k^{th} nearest neighbor, average distance to the k nearest neighbors, the inverse of the number of neighbors within a specific distance. Nevertheless, these metrics are not applicable to data streams.

C. Incremental Clustering Methods

Most clustering algorithms we have discussed earlier in this section assume a complete and static dataset to operate. However, new data becomes continuously available in many applications such as the data streams. With the aforementioned classical clustering algorithms, reclustering from scratch to account for data updates is too costly and inefficient. It is highly desired that the data can be processed and clustered in an incremental fashion. The recent representative clustering algorithms having mechanisms to handle data updates are BIRCH*, STREAM and CluStream.

BIRCH* [45] is a framework for fast, scalable and incremental clustering algorithms. In the BIRCH* family of algorithms, objects are read from the databases sequentially and inserted into incrementally evolving clusters which are represented by generalized cluster features (CF*s), the condensed and summarized representation of clusters. A new objects reading from the databases is inserted into the closest cluster. BIRCH* organizes all clusters in an in-memory index, and height-balanced tree, called CF*-tree. For a new object, the search for an appropriate cluster requires time logarithmic in the number of the clusters to a linear scan. CF*s are efficient because: (1) they occupy much less space than the naive representation; (2) the calculation of inter-cluster and intra-cluster measurements using the CF* is much faster than calculations involving all objects in clusters. The purpose of the CF*-tree is to direct a new object to the cluster closest to it. The non-leaf and leaf entries function differently, non-leaf entries are used to guide new objects to appropriate leaf clusters, whereas leaf entries represent the dynamically evolving clusters. However, clustering of high-dimensional datasets has not been studied in BIRCH*. In addition, BIRCH* cannot perform well when the clusters are not spherical in shape due to the fact that it relies on spherical summarization to produce the clusters.

STREAM [74] considers the clustering of continuously arriving data, and provides a clustering algorithm superior to the commonly used k-means algorithm. STREAM assumes that the data actually arrives in chunks $X_1, X_2, \cdots, X_n$, each of which fits into main memory. The streaming algorithm is as follows. For each chunk i, STREAM first assigns weight to points in the chunks according to their respective appearance frequency in the chunks ensuring that each point appear only once. The STREAM clusters each chunk using procedure LOCALSEARCH. For each chunk, only k weighted cluster centers are retained and the whole chunk is discarded in order to free the memory for new chunks. Finally, LOCALSEARCH is applied to the weighted centers retained from $X_1, X_2, \cdots, X_n$, to obtain a set of (weighted) centers for the entire stream $X_1, X_2, \cdots, X_n$.

In order to find clusters in different time horizons (such as the last month, last year or last decade), a new clustering method for data stream, called CluStream, is proposed in [8]. This approach provides the user the flexibility to explore the nature of the evolution of the clusters over different time periods. In order to avoid bookkeeping the huge amount of information about the clustering results in different time horizons, CluStream divides the clustering process into an online micro-clustering component and an offine macro-clustering component. The micro-clustering phase mainly collects online the data statistics for clustering purpose. This process is not dependent on any user input such as the time horizon or the required granularity of the clustering process. The aim is to maintain statistics at a sufficiently high level of granularity so that it can be effectively used by the offline components of horizon-specific macro-clustering as well as evolution analysis. The micro-clusters generated by the algorithm serve as an intermediate statistical representation which can be maintained in an efficient way even for a data stream of large volume. The macro-clustering process does not work on the original data stream that may be very large in size. Instead, it uses the compactly stored summary statistics of the micro-clusters. Therefore, the micro-clustering phase is not subject to the one-pass constraint of data stream applications.

D. Advantages and Disadvantages of Outlier Detection for Data Streams

The methods discussed in this subsection can detect outliers from data streams. The incremental LOF method is able to deal with continuously arriving data, but it may face an explosion of space consumption. Moreover, the incremental LOF method is not able to find outliers in subspaces in an automatic manner. The outlier detection methods for sensor networks cannot find projected outliers either. Unlike the clustering methods that are only appropriate for static databases, BIRCH*, STREAM and CluStream go one step further and are able to handle incrementally the continuously arriving data. Nevertheless, they are designed to use all the features of data in detecting outliers and are difficult to detect projected outliers.

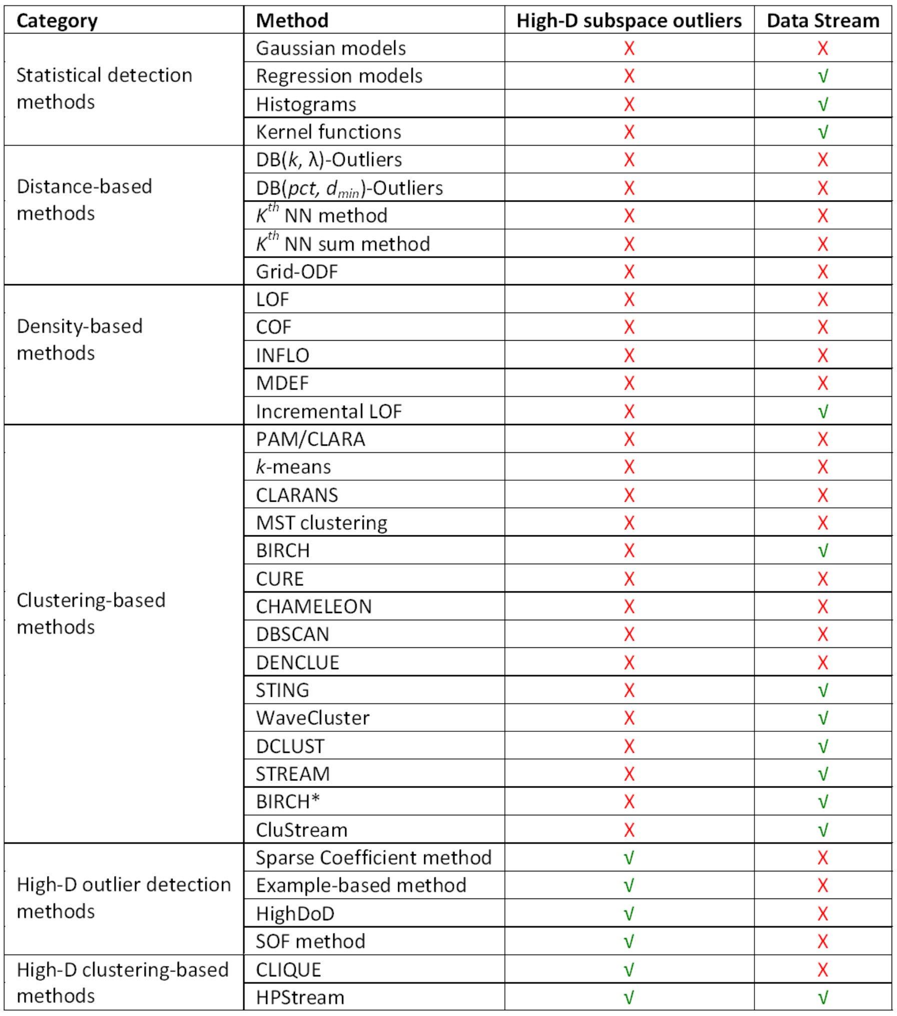
6.1. Summary

This section presents a comprehensive survey on the major existing methods for detecting point outliers from vector-like data sets. Both the conventional outlier detection methods that are mainly appropriate for relatively low dimensional static databases and the more recent methods that are able to deal with high-dimensional projected outliers or data stream applications have been discussed. For a big picture of these methods, we present a summary in Table 6. In this table, we evaluate each method against two criteria, namely whether it can detect projected outliers in a high-dimensional data space and whether it can handle data streams. The symbols of tick and cross in the table indicate respectively whether or not the corresponding method satisfies the evaluation criteria. From this table, we can see that the conventional outlier detection methods cannot detect projected outliers embedded in different subspaces; they detect outliers only in the full data space or a given subspace. Amongst these methods that can detect projected outliers, only HPStream can meet both criteria. However, being a clustering method, HPStream cannot provide satisfactory support for projected outliers detection from high-dimensional data streams.

7. Conclusions

In this paper, a comprehensive survey is presented to review the existing methods for detecting point outliers from various kinds of vector-like datasets. The outlier detection techniques that are primarily suitable for relatively low-dimensional static data, which serve the technical foundation for many of the methods proposed later, are reviewed first. We have also reviewed some of recent advancements in outlier detection for dealing with more complex high-dimensional static data and data streams.

It is important to be aware of the limitation of this survey. As it has clearly stated in Section 2, we only focus on the point outlier detection methods from vector-like datasets due to the space limit. Also, outlier detection is a fast developing field of research and more new methods will quickly emerge in the foreseeable near future. Driven by their emergence, it is believed that outlier detection techniques will play an increasingly important role in various practical applications where they can be applied to.



Category	Method	High-D subspace outliers	Data Stream
Statistical detection methods	Gaussian models	X	X
	Regression models	X	√
	Histograms	X	√
	Kernel functions	X	√
Distance-based methods	DB(k, λ)-Outliers	X	X
	DB(pct, d_{min})-Outliers	X	X
	K^{th} NN method	X	X
	K^{th} NN sum method	X	X
	Grid-ODF	X	X
Density-based methods	LOF	X	X
	COF	X	X
	INFLO	X	X
	MDEF	X	X
	Incremental LOF	X	√
Clustering-based methods	PAM/CLARA	X	X
	k-means	X	X
	CLARANS	X	X
	MST clustering	X	X
	BIRCH	X	√
	CURE	X	X
	CHAMELEON	X	X
	DBSCAN	X	X
	DENCLUE	X	X
	STING	X	√
	WaveCluster	X	√
	DCLUST	X	√
	STREAM	X	√
	BIRCH*	X	√
	CluStream	X	√
High-D outlier detection methods	Sparse Coefficient method	√	X
	Example-based method	√	X
	HighDoD	√	X
	SOF method	√	X
High-D clustering-based methods	CLIQUE	√	X
	HPStream	√	√

Figure 6. A summary of major existing outlier detection methods

References

[1] C. C. Aggarwal. On Abnormality Detection in Spuriously Populated Data Streams. SIAM International Conference on Data Mining (SDM'05), Newport Beach, CA, 2005.

[2] B. Abraham and G. E. P. Box. Bayesian analysis of some outlier problems in time series. *Biometrika* 66, 2, 229-236, 1979.

[3] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein an J. Widom. STREAM: The Stanford Stream Data Manager, *SIGMOD'03*, 2003.

[4] B. Abraham and A. Chuang. Outlier detection and time series modeling. *Technometrics* 31, 2, 241-248, 1989.

[5] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation intrusion detection expert system (nides), software users manual, beta-update release. *Technical Report*, Computer Science Laboratory, SRI International, 1994.

[6] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. of *1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pp 94-105, 1998.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data Mining Application. In proceeding of *ACM SIGMOD'99*, Philadelphia, PA, USA, 1999.

[8] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A Framework for Clustering Evolving Data Streams. In Proc. of *29th Very Large Data Bases (VLDB'03)*,pp 81-92, Berlin, Germany, 2003.

[9] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In Proc. of *30th Very Large Data Bases (VLDB'04)*, pp 852-863, Toronto, Canada, 2004.

[10] F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In Proc. of *6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*,Helsinki, Finland, pp 15-26, 2002.

[11] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu and J. S. Park. Fast algorithms for projected clustering. In Proc. of *1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pp 61-72, 1999.

[12] D. Anderson, A. Tamaru, and A. Valdes. Detecting unusual program behavior using the statistical components of NIDES. *Technical Report*, Computer Science Laboratory, SRI International, 1995.

[13] C. C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pp 70-81, 2000.

[14] C. C. Aggarwal and P. S. Yu. Outlier Detection in High Dimensional Data. In Proc. of *2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, Santa Barbara, California, USA, 2001.

[15] Charu C. Aggarwal and Philip S. Yu. 2005. An effective and efficient algorithm for high-dimensional outlier detection. *VLDB Journal*, 14: 211-221, Springer-Verlag Publisher.

[16] V. Barnett. The ordering of multivariate data (with discussion). *Journal of the Royal Statistical Society*. Series A 139, 318-354, 1976.

[17] C. Bishop. Novelty detection and neural network validation. In Proceedings of *IEEE Vision, Image and Signal Processing*, Vol. 141. 217-222, 1994.

[18] R. J. Beckman and R. D. Cook. Outliers. *Technometrics* 25, 2, 119-149, 1983.

[19] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft. When is nearest neighbors meaningful? In Proc. of *7th International Conference on Database Theory (ICDT'99)*, pp 217-235, Jerusalem, Israel, 1999.

[20] M. M. Breunig, H-P Kriegel, R. T. Ng and J. Sander. OPTICS-OF: Identifying Local Outliers. *PKDD'99*, 262-270, 1999.

[21] M. Breuning, H-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, pp 93-104, 2000.

[22] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: an efficient and robust access method for points and rectangles. In Proc. of *1990 ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, pp 322-331, Atlantic City, NJ,1990.

[23] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley, 3rd edition, 1994.

[24] L. Boudjeloud and F. Poulet. Visual Interactive Evolutionary Algorithm for High Dimensional Data Clustering and Outlier Detection. In Proc. of *9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'05)*, Hanoi, Vietnam, pp426-431, 2005.

[25] Branch, J. Szymanski, B. Giannella, C. Ran Wolff Kargupta, H. n-Network Outlier Detection in Wireless Sensor Networks. In Proc. of. *26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[26] H. Cui. Online Outlier Detection Detection Over Data Streams. *Master thesis*, Simon Fraser University, 2002.

[27] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD'04*, pages 112-124, 2004.

[28] V. Chandola, A. Banerjee, and V. Kumar. Outlier Detection-A Survey, *Technical Report*, TR 07-017, Department of Computer Science and Engineering, University of Minnesota, 2007.

[29] C. Chow and D. Y. Yeung. Parzen-window network intrusion detectors. In Proceedings of *the 16th International Conference on Pattern Recognition*, Vol. 4, Washington, DC, USA, 40385, 2002.

[30] D. E. Denning. An intrusion detection model. *IEEE Transactions of Software Engineering* 13, 2, 222-232, 1987.

[31] M. Desforges, P. Jacob, and J. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. In Proceedings of *Institute of Mechanical Engineers*, Vol. 212. 687-703, 1998.

[32] D. Dasgupta and F. Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In Proceedings of *the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1. Nashville, TN, 125-130, 2000.

[33] K. Das and J. G. Schneider: Detecting anomalous records in categorical datasets. *KDD'07*, 220-229, 2007.

[34] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In Proceedings of *the Seventeenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc., 2000.

[35] D. Endler. Intrusion detection: Applying machine learning to solaris audit data. In Proceedings of the 1*4th Annual Computer Security Applications Conference*, 268, 1998.

[36] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. *Applications of Data Mining in Computer Security*, 2002.

[37] M. Ester, H-P Kriegel, J. Sander, and X.Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In proceedings of *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Portland, Oregon, USA, 1996.

[38] E. Eskinand and S. Stolfo. Modeling system call for intrusion detection using dynamic window sizes. In Proceedings of *DARPA Information Survivability Conference and Exposition*, 2001.

[39] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, Series B (Methodological) 34, 3, 350-363, 1972.

[40] T. Fawcett. and F. Provost. Activity monitoring: noticing interesting changes in behavior. In Proceedings of the 5th *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 53-62, 1999.

[41] D. Freedman, R. Pisani and R. Purves. *Statistics*, W. W. Norton, New York, 1978.

[42] F. Grubbs Procedures for detecting outlying observations in samples. *Technometrics* 11, 1, 1-21, 1969.

[43] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[44] S. Guttormsson, R. M. II, and M. El-Sharkawi. Elliptical novelty grouping for on-line short-turn detection of

excited running rotors. *IEEE Transactions on Energy Conversion* 14, 1, 1999.

[45] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering Large Datasets in Arbitrary Metric Spaces. In Proc.s of *the 15th International Conference on Data Engineering (ICDE'99)*, Sydney, Australia, 1999.

[46] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In Proceedings of *the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, WA, USA, 1998.

[47] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[48] P. Helman and J. Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. In *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 27, 449-466, 1997.

[49] P. S. Horn, L. Feng, Y. Li, and A. J. Pesce. Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry* 47, 12, 2137-2145, 2001.

[50] J. Han and M Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2000.

[51] A. Hinneburg, and D.A. Keim. An Efficient Approach to Cluster in Large Multimedia Databases with Noise. *KDD'98*, 1998.

[52] W. Jin, A. K. H. Tung and J. Han. Finding Top n Local Outliers in Large Database. In Proc. of *7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'01)*, San Francisco, CA, pp 293-298, 2001.

[53] W. Jin, A. K. H. Tung, J. Han and W. Wang: Ranking Outliers Using Symmetric Neighborhood Relationship. *PAKDD'06*, 577-593, 2006.

[54] H. S. Javitz and A. Valdes. The SRI IDES statistical anomaly detector. In Proceedings of *the 1991 IEEE Symposium on Research in Security and Privacy*, 1991.

[55] G. Karypis, E-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer*, 32, Pages 68-75, 1999.

[56] E. M. Knorr and R. T. Ng. A unified approach for mining outliers. *CASCON'97*, 11, 1997.

[57] E. M. Knorr and R. T. Ng. A Unified Notion of Outliers: Properties and Computation. *KDD'97*, 219-222, 1997.

[58] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-based Outliers in Large Dataset. In Proc. of *24th International Conference on Very Large Data Bases (VLDB'98)*, New York, NY, pp 392-403, 1998.

[59] E. M. Knorr and R. T. Ng (1999). Finding Intentional Knowledge of Distance-based Outliers. In Proc. of *25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, pp 211-222, 1999.

[60] E. M. Knorr, R. T. Ng and V. Tucakov. Distance-Based Outliers: Algorithms and Applications. *VLDB Journal*, 8(3-4): 237-253, 2000.

[61] T. M. Khoshgoftaar, S. V. Nath, and S. Zhong. Intrusion Detection in Wireless Networks using Clusterings Techniques with Expert Analysis. Proceedings of *the Fourth International Conference on Machine Leaning and Applications (ICMLA'05)*, Los Angeles, CA, USA, 2005.

[62] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John wiley&Sons, 1990.

[63] C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In Proceedings of *the 2002 ACM Symposium on Applied computing*, 201-208, 2002.

[64] X. Li and J. Han: Mining Approximate Top-K Subspace Anomalies in Multi-Dimensional Time-Series Data. *VLDB*, 447-458, 2007.

[65] J. Lee, J. Han and X. Li. Trajectory Outlier Detection: A Partition-and-Detect Framework. *ICDE'08*, 140-149, 2008.

[66] J. Laurikkala, M. Juhola1, and E. Kentala. 2000. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 20-24, 2000.

[67] G. Manson. Identifying damage sensitive, environment insensitive features for damage detection. In Proceedings of *the IES Conference*. Swansea, UK, 2002.

[68] J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proc. of *5th Berkeley Symp. Math. Statist, Prob.*, 1, pages 281-297, 1967.

[69] M. V. Mahoney and P. K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In Proceedings of *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 376-385, 2002.

[70] G. Manson, G. Pierce, and K. Worden. On the long-term stability of normal condition for damage detection in a composite panel. In Proceedings of *the 4th International Conference on Damage Assessment of Structures*, Cardiff, UK, 2001.

[71] G. Manson, S. G. Pierce, K. Worden, T. Monnier, P. Guy, and K. Atherton. Long-term stability of normal condition data for novelty detection. In Proceedings of *Smart Structures and Integrated Systems*, 323-334, 2000.

[72] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD'03*, pages 631-636, 2003.

[73] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In proceedings of *the 20th VLDB Conference*, pages 144-155, 1994.

[74] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-Data Algorithms For High-Quality Clustering. In Proceedings of *the 18th International Conference on Data Engineering (ICDE'02)*, San Jose, California, USA, 2002.

[75] M. I. Petrovskiy. Outlier Detection Algorithms in Data Mining Systems. *Programming and Computer Software*, Vol. 29, No. 4, pp 228-237, 2003.

[76] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics* 33, 1065-1076, 1962.

[77] S. Papadimitriou, H. Kitagawa, P. B. Gibbons and C. Faloutsos: LOCI: Fast Outlier Detection Using the Local Correlation Integral. *ICDE'03*, 315, 2003.

[78] D. Pokrajac, A. Lazarevic, L. Latecki. Incremental Local Outlier Detection for Data Streams, *IEEE symposiums on computational Intelligence and Data Mining (CIDM'07)*, 504-515, Honolulu, Hawaii, USA, 2007.

[79] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In Proceedings of *20th NIST-NCSC National Information Systems Security Conference*, 353-365, 1997.

[80] T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Record* 32(4): 77-82, 2003.

[81] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, pp 427-438, 2000.

[82] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Wavelet based Clustering Approach for Spatial Data in Very Large Database. *VLDB Journal*, vol.8 (3-4), pages 289-304, 1999.

[83] H. E. Solberg and A. Lahti. Detection of outliers in reference distributions: Performance of horn's algorithm. *Clinical Chemistry* 51, 12, 2326-2332, 2005.

[84] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos. Neighborhood Formation and Anomaly Detection in Bipartite Graphs. *ICDM'05*, 418-425, 2005.

[85] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos. Relevance search and anomaly detection in bipartite graphs. *SIGKDD Explorations* 7(2): 48-55, 2005.

[86] L. Tarassenko. Novelty detection for the identification of masses in mammograms. In Proceedings of *the 4th IEEE International Conference on Artificial Neural Networks*, Vol. 4. Cambridge, UK, 442-447, 1995.

[87] J. Tang, Z. Chen, A. Fu, and D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In Proc. of *6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*, Taipei, Taiwan, 2002.

[88] W. Wang, J. Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In Proceedings of *23rd VLDB Conference*, pages 186-195, Athens, Green, 1997.

[89] W. Wang, J. Zhang and H. Wang. Grid-ODF: Detecting Outliers Effectively and Efficiently in Large Multi-dimensional Databases. In Proc. of *2005 International Conference on Computational Intelligence and Security (CIS'05)*, pp 765-770, Xi'an, China, 2005.

[90] K. Yamanishi and J. I. Takeuchi. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In Proceedings of *the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 389-394, 2001.

[91] K. Yamanishi, J. I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery* 8, 275-300, 2004.

[92] C.T. Zahn. Graph-theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transaction on Computing*, C-20, pages 68-86, 1971.

[93] J. Zhang, Q. Gao and H. Wang. Outlying Subspace Detection in High dimensional Space. *Encyclopedia of Database Technologies and Applications (2nd Edition)*, Idea Group Publisher, 2009.

[94] J. Zhang and H. Wang. Detecting Outlying Subspaces for High-dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems: An International Journal (KAIS)*, Springer-Verlag Publisher, 2006.

[95] J. Zhang, W. Hsu and M. L. Lee. Clustering in Dynamic Spatial Databases. *Journal of Intelligent Information Systems (JIIS)* 24(1): 5-27, Kluwer Academic Publisher, 2005.

[96] C. Zhu, H. Kitagawa and C. Faloutsos. Example-Based Robust Outlier Detection in High Dimensional Datasets. In Proc. of *2005 IEEE International Conference on Data Management(ICDM'05)*, pp 829-832, 2005.

[97] C. Zhu, H. Kitagawa, S. Papadimitriou and C. Faloutsos. OBE: Outlier by Example. *PAKDD'04*, 222-234, 2004.

[98] S. Zhong, T. M. Khoshgoftaar, and S. V. Nath. A clustering approach to wireless network intrusion detection. In *ICTAI*, pages 190-196, 2005.

[99] J. Zhang, M. Lou, T. W. Ling and H. Wang. HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data. In Proc. of *30th International Conference on Very Large Data Bases (VLDB'04)*, demo, pages 1265-1268, Toronto, Canada, 2004.

[100] J. Zhang, Q. Gao and H. Wang. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *2006 IEEE International Conference on Data Mining (ICDM'06)*, pages 731-740, Hong Kong, China, 2006.

[101] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In proceedings of *the 1996 ACM International Conference on Management of Data (SIGMOD'96)*, pages 103-114, Montreal, Canada, 1996.

[102] J. Zhang and H. Wang. Detecting Outlying Subspaces for High-dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems (KAIS)*, 333-355, 2006.

[103] J. Zhang, Q. Gao, H. Wang, Q. Liu, K. Xu. Detecting Projected Outliers in High-Dimensional Data Streams. DEXA 2009: 629-644, 2009.