Outlier Detection in Clustering

Svetlana Cherednichenko

24.01.2005

University of Joensuu
Department of Computer Science
Master's Thesis

TABLE OF CONTENTS

List of Figures

List of Tables

Abstract

Outlier detection is a fundamental issue in data mining, specifically it has been used to detect and remove anomalous objects from data. Outliers arise due to mechanical faults, changes in system behaviour, fraudulent behaviour, network intrusions or human errors.

Firstly, this thesis presents a theoretical overview of outlier detection approaches. A novel outlier detection method is proposed and analyzed, it is called *Clustering Outlier Removal* (COR) algorithm. It provides efficient outlier detection and data clustering capabilities in the presence of outliers, and based on filtering of the data after clustering process. The algorithm of our outlier detection method is divided into two stages. The first stage provides *k*-means process. The main objective of the second stage is an iterative removal of objects, which are far away from their cluster centroids. The removal occurs according to a chosen threshold. Finally, we provide experimental results from the application of our algorithm on a KDD Cup1999 datasets to show its effectiveness and usefulness. The empirical results indicate that the proposed method was successful in detecting intrusions and promising in practice. We also compare COR algorithm with other available methods to show its important advantage against existing algorithms in outlier detection.

KEY WORDS: outlier detection, clustering, intrusions.

Acknowledgments

I am particularly grateful to my senior supervisor, Professor Pasi Fränti, for his helpful and constructive comments, for his guidance during my study. I would like to express my thanks to Ville Hautamäki for support and assistance. Also, I would like to express my gratitude to Ismo Kärkkäinen for providing information. I wish to thank my friends for their encouragement. And most importantly, this has been a great work that would not have possible without the moral support of my parents. I truly appreciate their help.

1. INTRODUCTION

1.1. Basic definitions

Data mining, in general, deals with the discovery of non-trivial, hidden and interesting knowledge from different types of data. With the development of information technologies, the number of databases, as well as their dimension and complexity, grow rapidly. It is necessary what we need automated analysis of great amount of information. The analysis results are then used for making a decision by a human or program. One of the basic problems of data mining is the *outlier* detection.

An outlier is an observation of the data that deviates from other observations so much that it arouses suspicions that it was generated by a different mechanism from the most part of data [41]. Inlier, on the other hand, is defined as an observation that is explained by underlying probability density function. This function represents probability distribution of main part of data observations [17].

Outliers may be erroneous or real in the following sense. Real outliers are observations whose actual values are very different than those observed for the rest of the data and violate plausible relationships among variables. Erroneous outliers are observations that are distorted due to misreporting or misrecording errors in the data-collection process. Outliers of either type may exert undue influence on the results of statistical analysis, so they should be identified using reliable detection methods prior to performing data analysis [47].

Many data-mining algorithms find outliers as a side-product of clustering algorithms. However these techniques define outliers as points, which do not lie in clusters. Thus, the techniques implicitly define outliers as the background noise in which the clusters are embedded. Another class of techniques defines outliers as points, which are neither a part of a cluster nor a part of the background noise; rather they are specifically points which behave very differently from the norm [2].

Typically, the problem of detecting outliers has been studied in the statistics community. The user has to model the data points using a statistical distribution, and points are determined to be outliers depending on how they appear in relation to the postulated model. The main problem with these approaches is that in a number of situations, the user might simply not have enough knowledge about the underlying data distribution [36].

Outliers can often be individuals or groups of clients exhibiting behavior outside the range of what is considered normal. Outliers can be removed or considered separately in *regression modeling* to improve accuracy which can be considered as benefit of outliers. Identifying them prior to modeling and analysis is important [41]. The regression modeling consists in finding a dependence of one random variable or a group of variables on another variable or a group of variables.

In the context of outlier-based association method, outliers are observations markedly different from other points. When a group of points have some common characteristics, and these common characteristics are “outliers”, these points are associated [29].

Almost all studies that consider outlier identification as their primary objective are in statistics. The test depends on the distribution; whether or not the distribution parameters are known; the number of excepted outliers; the types of excepted outliers [25].

1.2. Practical applications

The identification of an outlier is affected by various factors, many of which are of interest for practical applications. For example, fraud, or criminal deception, will always be a costly problem for many profit organizations. Data mining can minimize some of these losses by making use of the massive collections of customer data [35]. Using web log files becomes possible to recognize fraudulent behavior, changes in behavior of customers or faults in systems. Outliers arise by reasons of such incidents. Thus typical fault detection can discover exceptions in the amount of money spent, type of items purchased, time and location. Many fraud cases can happen, for example, if someone has your name, credit card number, expiration date and billing address. All this information is very easy to obtain even from your home mailbox or any on-line transaction that you had before [4]. So, automatic systems for preventing fraudulent use of credit cards detect unusual transactions and may block such transactions on earlier stages.

Another example is a computer security intrusion detection system, which finds outlier patterns as a possible intrusion attempts. Intrusion detection corresponds to a suite of techniques that are used to identify attacks against computers and network infrastructures. Anomaly detection is a key element of intrusion detection in which perturbations of normal behavior suggest the presence of intentionally or unintentionally induced attacks, faults and defects [27]. Detecting outliers has practical application in more wide spheres: pharmaceutical research, weather prediction, financial applications, marketing and customer segmentation.

The system applied to real network traffic data is illustrated in Figure 1. The basic steps consist of converting data, building detection model, analysis and summarizing of results.

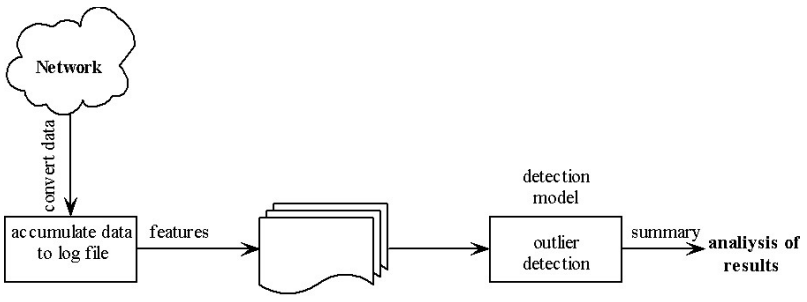

Figure 1. Outlier detection process in Data Mining.

In handwritten word recognition some errors were caused by non-character images that were assigned high character confidence value [32]. Segmentation and dynamic

programming (DP)-based approaches are used for outlier rejection in off-line handwritten word recognition method. The flow diagram is shown in Figure 2. Segmentation splits a word image into partial characters than use character classifier and DP to obtain the optimal segmentation and recognition result. The recognition process assigns a match score to each candidate string and the highest score determines the result. The focus of this approach is to assign low character confidence values to non-character images, which means to reject outlier. The neural networks were used to realize outlier rejection, where valid patterns only activate the output node corresponding to the class, which the pattern belongs to. Outliers do not activate any output node [32].

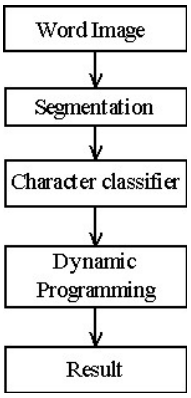

Figure 2. Diagram of handwritten word recognition system.

1.3. Outliers in clustering

The outlier detection problem in some cases is similar to the classification problem. For example, the main concern of *clustering-based* outlier detection algorithms is to find *clusters* and outliers, which are often regarded as noise that should be removed in order to make more reliable clustering [17]. Some noisy points may be far away from the data points, whereas the others may be close. The far away noisy points would affect the result more significantly because they are more different from the data points. It is desirable to identify and remove the outliers, which are far away from all the other points in cluster [20]. So, to improve the clustering such algorithms use the same process and functionality to solve both clustering and outlier discovery [17].

1.4. Purpose of this research

In this work, we consider outliers defined as points, which are far from the most of other data. The purpose of proposed approach is first to apply *k-means algorithm* and then find outliers from the resulting clusters. After that again apply k-means, and so on until the number of points will not be changed in dataset. The principle of outliers removal depends on the threshold and the distortion. Threshold is set by user and distortion defined as the ratio of distance for nearest point to the cluster *centroid* divided by distance of furthest

point in the same *partition*. If the distortion is less than the threshold, this furthest point is considered to be outlier for this cluster. So, we propose a *clustering-based* technique to identify outliers and simultaneously produce data clustering. Our outlier detection process at the same time is effective for extracting clusters and very efficient in finding outliers.

1.5. Organization of the thesis

The rest of the thesis is structured as follows. In Section 2, we present definition of clustering and general classification of clustering algorithms. In Section 3, we consider outlier detection methods in details, given classification by distribution-based, distance-based and density-based approaches. In Section 4 we give detailed description of the new method. Experimental results are reported on the real KDD Cup 1999 data to show the performance of new algorithm in Section 5 and comparison to other outlier detection methods is presented. Concluding remarks are given in Section 6.

2. CLUSTERING

2.1. Notations of terms

In this section we formally define the notations used in the reminder of thesis.

N Number of data objects.
M Number of clusters.
K Number of attributes.
X Set of N data objects $X = \{x_1, x_2, ..., x_N\}$.
P Set of N cluster indices $P = \{p_1, p_2, \ldots, p_N\}$.
C Set of M cluster representatives $C = \{c_1, c_2, \ldots, c_M\}$.

2.2. Problem definition

Clustering, or *unsupervised classification*, will be considered as a combination problem where the aim is to partition a set of *data object* into a predefined number of clusters. Number of clusters might be found by means of the *cluster validity criterion* or defined by user. Data object, *feature vector* and *attribute* are shown in Figure 3. The attributes of an object can be represented by a feature vector, where each element of the vector corresponds to one attribute. There are no examples that what kind of desirable relations should be valid among the data and that is why clustering is perceived as an unsupervised process. The objects with similar features should be grouped together and objects with different features placed in separate groups [10]. Dissimilarities are assessed based on the attribute values describing the objects. Often, distance measure between the two feature vectors is used to show dissimilarity between objects [45].

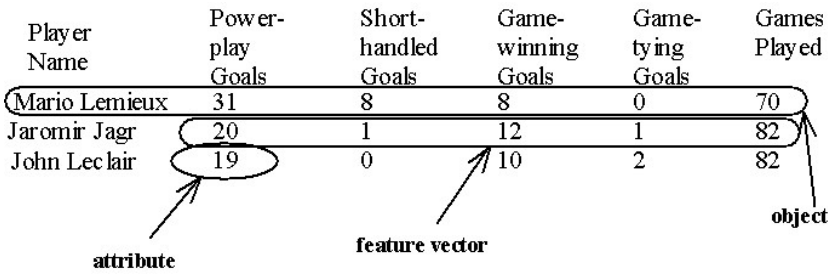

Figure 3. Explanations for basic concepts.

2.3. Clustering application

Clustering problems are widely used in numerous applications, such as customer segmentation, classification, and trend analysis. For example, consider a retail database records containing items purchased by customers. A clustering procedure could group the customers in such a way that customers with similar buying patterns are in the same cluster [13]. Many real-word applications deal with high dimensional data. It has always been a challenge for clustering algorithms because of the manual processing is practically impossible [3]. A high quality computer-based clustering removes the unimportant features and replaces the original set by a smaller representative set of data objects. As a result, the size of data reduces and, therefore, cluster analysis can contribute in compression of the information included in data. Cluster analysis is applied for prediction. Suppose, for example, that the cluster analysis is applied to a dataset concerning patients infected by the same disease. The result is a number of clusters of patients, according to their reaction to specific drugs. So, for a new patient, we identify the cluster in which he can be classified and based on this decision his medication can be made [13].

2.4. Clustering problems

The general clustering problem includes three subproblems: (i) selection of the evaluation function; (ii) decision of the number of groups in the clustering; and (iii) the choice of the clustering algorithm [10].

2.4.1. Evaluation of clustering

An *objective function* is used for evaluation of clustering methods. The choice of the function depends upon the application, and there is no universal solution of which measure should be used. Commonly used a basic objective function is defined as (2.1):

$$f(P,C) = \sum_{i=1}^{N} d(x_i, c_{p_i})^2 , \tag{2.1}$$

where P is partition and C is the cluster representatives, d is a distance function. The Euclidean distance and Manhattan distance are well-known methods for distance measurement, which are used in clustering context. Euclidean distance is expressed as (2.2):

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{K} (x_1^i - x_2^i)^2} \tag{2.2}$$

and Manhattan distance is calculated as (2.3):

$$d(x_1, x_2) = \sum_{i=1}^{K} \left| x_1^i - x_2^i \right| . \tag{2.3}$$

2.4.2. Number of clusters

The choice of number of the clusters is an important subproblem of clustering. Since a priori knowledge is generally not available and the vectors dimensions are often higher than two, which do not have visually apparent clusters. The solution of this problem directly affects the quality of the result. If the number of clusters is too small, different objects in data will not be separated. Moreover, if this estimated number is too large, relatively regions may be separated into a number of smaller regions [46]. Both of these situations are to be avoided. This problem is known as the cluster validation problem. The aim is to estimate the number of clusters during the clustering process. The basic idea is the evaluation of a clustering structure by generating several clustering for various number of clusters and compare them against some evaluation criteria. In general, there are three approaches to investigate cluster validity [14]. In *external* approach, the clustering result can be compared to an independent partition of the data built according to our intuition of the structure of the dataset. The *internal criteria* approach uses some quantities or features inherent in the dataset to evaluate the result. The basic idea of the third approach, *relative criteria*, is the evaluation of a clustering structure by comparing it to other clustering schemes, produced by the same algorithm but with different input parameter values. The two first approaches are based on statistical tests and their major drawback is their high computational cost. In the third approach aim is to find the best clustering scheme that a clustering algorithm can define under certain assumptions and parameters. More information about clustering validity methods you can find in [14], [15].

2.5. Classification of methods

Clustering algorithms can be classified according to the method adopted to define the individual clusters. The algorithms can be broadly classified into the following types: *partitional clustering*, *hierarchical clustering*, *density-based clustering* and *grid-based clustering* [33]. These algorithms are based on distance measure between two objects. Basically the goal is to minimize the distance of every object from the center of the cluster to which the object belongs.

2.5.1. Partitional clustering

Partition-based methods construct the clusters by creating various partitions of the dataset. So, partition gives for each data object the cluster index p_i. The user provides the desired number of clusters M, and some criterion function is used in order to evaluate the proposed partition or the solution. This measure of quality could be the average distance between clusters; for instance, some well-known algorithms under this category are *k-means*, *PAM* and *CLARA* [23], [48]. One of the most popular and widely studied clustering methods for objects in Euclidean space is called *k-means clustering*. Given a set of N data objects x_i and an integer M number of clusters. The problem is to determine C, which is a set of M cluster representatives c_j, as to minimize the mean squared Euclidean distance from each data object to its nearest centroid.

The algorithm starts with an initial solution and then involves an iterative scheme that operates over a fixed number of clusters, while a stopping criterion is met, i.e. the centers of the clusters stop changing.

Algorithm contains simple steps as follows. Firstly, initial solution is assigned to random to the *M* sets:

$$c_j \leftarrow x_i \,\big|\, j = random(1, M), i = random(1, N).$$

Then, in the first step, the data objects are partitioned as to each cluster centroid is closest to the data object in respect to the distance function:

$$p_i \leftarrow \arg\min_{1 \le j \le M} d(x_i, c_j)^2 \;\forall i \in [1, N].$$

In the second step, the cluster centroids are recalculated corresponding to the new partition:

$$c_j \leftarrow \frac{\sum_{p_i = j} x_i}{\sum_{p_i = j} 1} \forall\; j \in [1, M].$$

The number of iterations depends upon the dataset, and upon the quality of initial clustering data. The *k*-means algorithm is very simple and reasonably effective in most cases. Completely different final clusters can arise from differences in the initial randomly chosen cluster centers. In final clusters *k*-means do not represent global minimum and it gets as a result the first local minimum. Main advantage of the k-means method in follows: almost any solution not obtained by a *k*-means method can be improved. Disadvantage is that these methods only work well for finding clusters with spherical shapes and similar sizes.

2.5.2. Hierarchical clustering

Hierarchical clustering methods build a cluster hierarchy, i.e. a tree of clusters also known as *dendogram*. A dendrogram is a *tree diagram* often used to represent the results of a cluster analysis. Hierarchical clustering methods are categorized into *agglomerative* (bottom-up) and *divisive* (top-down) as shown in Figure 4. An agglomerative clustering starts with one-point clusters and recursively merges two or more most appropriate clusters. In contrast, a divisive clustering starts with one cluster of all data points and recursively splits into nonoverlapping clusters.

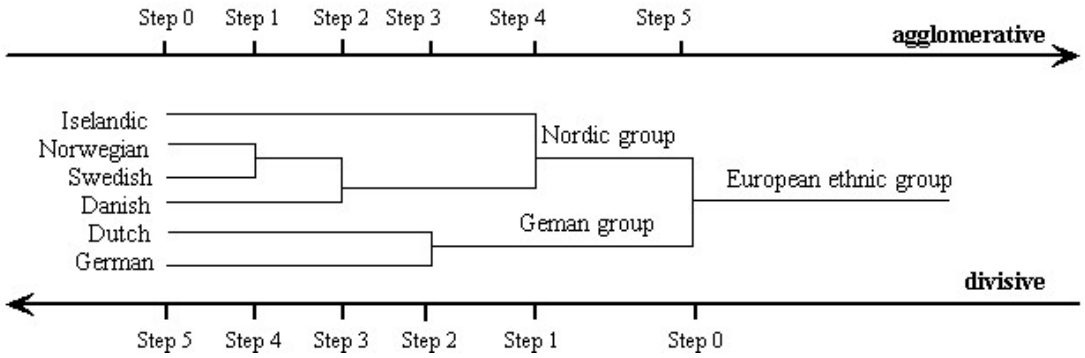

Figure 4. Example of dendogram.

The process continues until a stopping criterion (frequently, the requested number *M* of clusters) is achieved. Hierarchical methods provide ease of handling of any form of similarity or distance, because use distance matrix as clustering criteria. However, most hierarchical algorithms do not improve intermediate clusters after their construction. Furthermore, the termination condition has to be specified. Hierarchical clustering algorithms include *BIRCH* [7] and *CURE* [11].

2.5.3. Density-based and grid-based clustering

The key idea of density-based methods is that for each object of a cluster the neighborhood of a given radius has to contain a certain number of objects; i. e. the density in the neighborhood has to exceed some threshold. The shape of a neighborhood is determined by the choice of a distance function for two objects. These algorithms can efficiently separate noise [9]. *DBSCAN* [5] and *DBCLASD* [42] are the well-known methods in the density-based category.

The basic concept of grid-based clustering algorithms is that they quantize the space into a finite number of cells that form a grid structure. And then these algorithms do all the operations on the quantized space. The main advantage of the approach is its fast processing time, which is typically independent of the number of objects, and depends only on the number of grid cells for each dimension [33]. Famous methods in this clustering category are *STING* [40] and *CLIQUE* [1].

Other techniques available include *model-based clustering*, *constraint-based* and *fuzzy clustering* [37]. Model-based methods hypothesize a model for each of the clusters and find the best fit of that model to each other. One method from this category is EM algorithm [22]. The idea of constraint-based clustering is finding clusters that satisfy user-specified constraints, for example as in *COD CLARANS* method [39]. Fuzzy clustering methods attempt to find the most characteristic objects in each cluster, which can be considered as the center of the cluster, and then, find the membership for each object in the cluster. A common fuzzy clustering algorithm is Fuzzy *C-Means* [13].

3. OUTLIER DETECTION METHODS

Most outlier detection techniques treat objects with K attributes as points in $\Re^K$ space and these techniques can be divided into three main categories. The first approach is *distance-based* methods, which distinguish potential outliers from others based on the number of objects in the neighborhood [19]. *Distribution-based* approach deals with *statistical methods* that are based on the probabilistic data model. A probabilistic model can be either a priori given or automatically constructed using given data. If the object does not suit the probabilistic model, it is considered to be an outlier [34]. Third, *density-based* approach detects local outliers based on the local density of an object's neighborhood [21]. These methods use different density estimation strategy. A low local density on the observation is an indication of a possible outlier [18].

3.1. Distance-based approach

3.1.1. Distance-based definitions for outliers

In *Distance-based* methods outlier is defined as an object that is at least d_{min} distance away from k percentage of objects in the dataset. The problem is then finding appropriate d_{min} and k such that outliers would be correctly detected with a small number of false detections. This process usually needs domain knowledge [18].

In the present section we define objects as points for simple interpretation and consider definitions as a special case of [18]. Firstly, consider the definition proposed by Knorr and Ng [36], which both a simple and intuitive:

> ***Definition:*** *A point x in a dataset is an outlier with respect to the parameters k and d, if no more than k points in the dataset are at a distance d or less from x.*

To explain the definition by example we take parameter $k = 3$ and distance d as shown in Figure 5. Here are points x_i and x_j be defined as outliers, because of inside the circle for each point lie no more than 3 other points. And x' is an inlier, because it has exceeded number of points inside the circle for given parameters k and d.

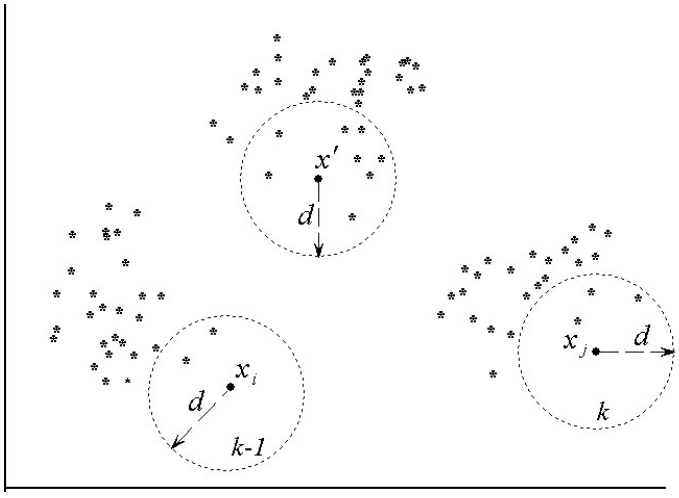

Figure 5. Illustration of outlier definition by Knorr and Ng.

This approach does not require any a priori knowledge of data distributions as the statistics methods do. However, this distance-based approach has certain shortcomings:

1. It requires the user to specify a distance *d*, which could be difficult to determine a-priori.
2. It does not provide a ranking for the outliers: for instance a point with a very few neighboring points within a distance *d* can be regarded in some sense as being a stronger outlier than a point with more neighbors within distance *d*.

It becomes increasingly difficult to estimate parameter *d* with increasing dimensionality. Thus, if one picks radius *d* slightly small, then all points are outliers. If one picks *d* slightly large, then no point is an outlier. So, user needs to pick *d* to a very high degree of accuracy in order to find a modest number of points, which can be defined as outliers [2].

Consider by example how can such kind of imperfections appear. Actually, in Figure 6, x_i and x_j are outliers, but radius d_1 is too large, hence inside the circles there are too many points. In this case we define x_i and x_j as incorrect inliers. Oppositely if the radius d_2 is too small, so inside the circles lie very small number of points, and then x_l and x_k be wrong outliers.

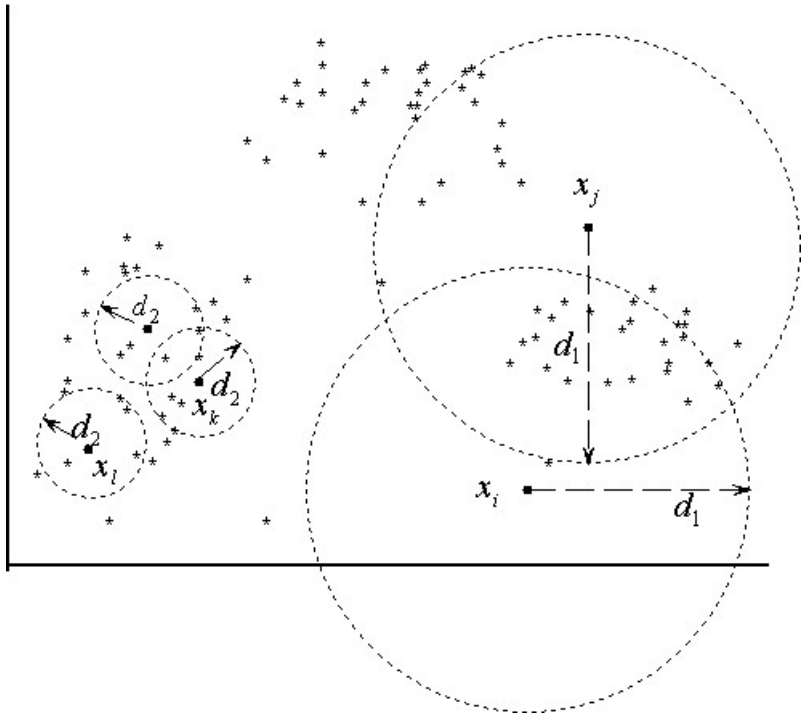

Figure 6. Illustration of definition shortcomings.

The next definition, proposed by Ramaswamy *et al.* [36], for outliers in the high dimensional data does not require user to specify the distance parameter *d*. Instead, it is based on the distance of the k^{th} nearest neighbor of a point. Let $D^k(x)$ use to denote the distance of point *x* from its k^{th} nearest neighbor and ranking points on the basis of their $D^k(x)$ distance, leading to the following definition for D_n^k .

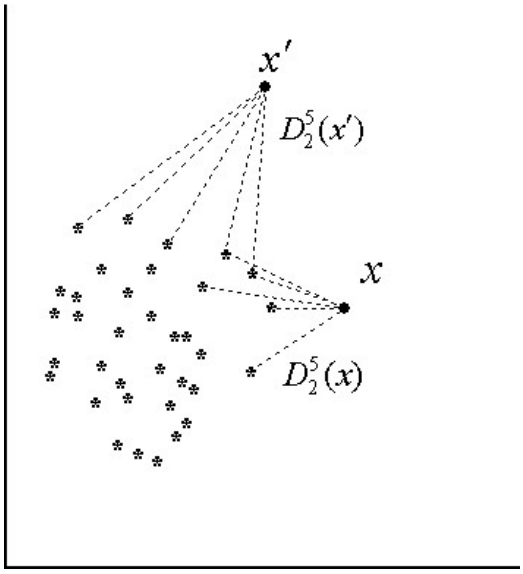

Figure 7. Illustration of outlier definition for high dimensional data without parameter *d.*

Definition: *Given an input dataset with N points, parameters n and k , a point x is a* D_n^k *outlier if there are no more than n-1 other points x' such that* $D^k(x')>D^k(x)$.

Intuitively, $D^k(x)$ is a measure of how much of an outlier point *x* is. For example, points with large values for $D^k(x)$ have more sparse neighborhoods. In this definition, user has to specify the number of outliers *n* that he wants to get. In other words, if points are ranking according to their $D^k(x)$ distance, the top *n* objects in this ranking are considered to be outliers. We can use any of the L_p metrics such as L_1 (Manhattan) or L_2 (Euclidean) metrics for measuring the distance between a pair of objects.

Consider one simple example of definition above. Let *k=5* and *n=2*. In Figure 7 two outliers represent by points *x* and *x′*. Visually we can observe what distance of point *x′* from 5 nearest neighbors $D^k(x')$ more than $D^k(x)$, that is distance of point *x* from its 5 nearest neighbors. So, we conclude, if we will take *n=3*, we will have *3* outliers.

3.1.2. Hybrid-random algorithm

Hybrid-random algorithm was developed in [25]. It uses *Donoho-Stahel Estimator* (DSE) for distance-based operations in high-dimensional database. If two similar attributes are being compared, and these attributes are independent and have the same scale and variability, then all objects within distance *d* of a object x_i lie within the circle of radius *d* centered at x_i, as shown in Figure 8 on the left. In the presence of different scales, variability, and correlation, all objects within distance *d* of a object x_i lie within an ellipse as in Figure 8 in the middle. If there is no correlation, then the major and minor axes of the ellipse lie on the standard coordinate axes but if there is correlation, then the ellipse is rotated through some angle θ, Figure 8 on the right.

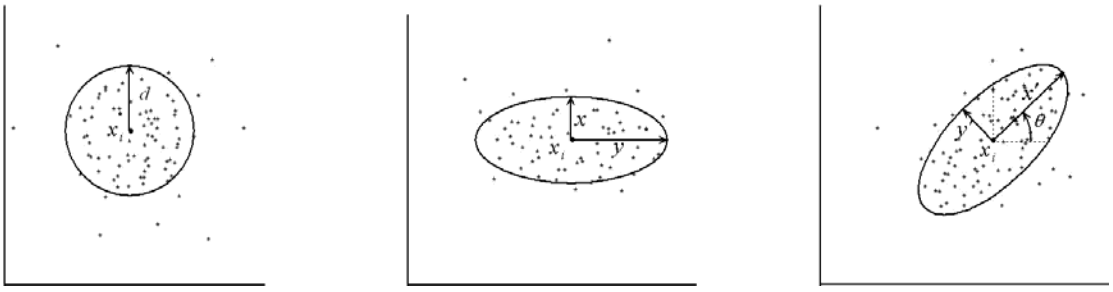

Figure 8. Data objects of the same scale and variability (left). Different scales, variability and no correlation (middle). Different scales, variability and correlation (right).

DSE is a robust space transformation, it is commonly used for comparing of different attributes, because they can have the different scales, units and variability. For example, blood pressure vs. body temperature, high variability for blood pressure vs. low variability for body temperature. Also attributes may be correlated, for instance age and blood pressure. At that rate, such attributes have to be normalized or standardized. But another solution is to use a robust space transformation.

DSE possesses two important properties. The first is the *Euclidean* property. It says that while inappropriate in the original space, the Euclidean distance function becomes reasonable in the DSE transformation space. The second, and more important, property is the *stability* property. It says that the transformed space is robust against updates, in other words the scale in space does not change even after modifications of the data.

In general, an estimator, which applies transformation, is also called a *scatter matrix*, it is a $K \times K$ square matrix, where K is the dimensionality of the original data space. DSE is defined through the *fixed-angle algorithm*. Short description of the algorithm include three steps: step 1 computes for each object and each angle θ, the degree of being outlier of the object with respect to θ. As a measure of how outlying each object is over all possible angles, step 2 computes, for each object, the maximum degree of outlyingness over all possible θ's. In step 3, if this maximum degree for a object is too high, the influence of this object is weakened by a decreasing weight function. Finally, with all objects weighted accordingly, the location center and the covariance matrix are computed.

In Hybrid-random algorithm subsumpling is first applied for a very small number of subsamples. Then from the fixed-angle algorithm, we know that projection vectors too close to each other do not give markedly different results.

Using the Euclidean inner product and Law of Cosines, a collision between two vectors a and b occurs if

$$dist^2(a,b) = \|a-b\|^2 = 2(1-|a^T b|) \le (2\delta)^2,$$

where δ is a radius of a patch on the surface of the K-d unit hypersphere [26].

3.1.3. A unified notion of outliers

As the distributions of the attribute values are almost unknown and no standard distribution can adequately model the observed distribution, so choice of suitable tests requires non-trivial computational effort for large datasets. We need a unified notion of outliers, in [25] it is defined as follows:

> ***Definition:*** *An object x_i in a dataset $\mathcal{X}$ is a UO(p, d) - outlier if at least fraction p of the objects in $\mathcal{X}$ are more than distance d from x_i.*

The term *UO(p, d)*-outlier is using as notation for a Unified Outlier with parameters *p* and *d*. The approach for finding all *UO(p, d)*-outliers relies on an cell structure. The idea is to reduce object-by-object processing to cell-by-cell processing [24].

In [25] is shown the cell structure for the two dimensional case, where the length of cell is $l = d/2\sqrt{2}$. Let $C_{x,y}$ denote the cell that is at the intersection of row x and column y. The Layer-1 $(\mathcal{L}_1)$ neighbors of $C_{x,y}$ are all the immediate neighboring cells of $C_{x,y}$ as defined in the usual sense, i.e.,

$$\mathcal{L}_1(C_{x,y}) = \{\, C_{u,v} \mid u = x \pm 1,\ v = y \pm 1,\ C_{u,v} \neq C_{x,y} \,\}.$$

The Layer-2 $(\mathcal{L}_2)$ neighbors of $C_{x,y}$ are all the cells within 3 cells of $C_{x,y}$, that is,

$$\mathcal{L}_2(C_{x,y}) = \{\, C_{u,v} \mid u = x \pm 3,\ v = y \pm 3,\ C_{u,v} \notin \mathcal{L}_1(C_{x,y}),\ C_{u,v} \neq C_{x,y} \,\}.$$

The general approach for computing outliers includes the following properties:

a) If there are more than k objects in $C_{x,y}$, none of the objects in $C_{x,y}$ is an outlier.

b) If there are more than k objects in $C_{x,y} \cup L_1(C_{x,y})$, none of the objects in $C_{x,y}$ is an outlier.

c) If there are less or equal to k objects in $C_{x,y} \cup L_1(C_{x,y}) \cup L_2(C_{x,y})$, every object in $C_{x,y}$ is an outlier.

Here k denotes the maximum number of objects that can be inside the d is the neighborhood of an outlier, i.e. $k=N(1-p)$. These properties help to identify outliers or non-outliers in a cell-by-cell manner rather on an object-by-object basis [25]. Algorithm *FindAllOuts* uses the cell structure as described above, is considered in [24].

3.1.4. Data association method

In the outlier-based association method, an outlier score function is defined to measure the extremeness of a cell. The more extreme a cell is, the higher outlier score it gets [29]. Here cell c is defined as a vector of the values of attributes with dimension t, where $t \leq K$. So a cell is a subset of object's attributes. For example, if the attributes are quantity, time, product and geography, then Sales, January 1994, Candy Bars and the United states will be a cell [30]. Since each object can also treated as a cell: $cell(x_i) = (x_1, x_2, \ldots, x_K)$.

The following rule is used to associated data: for two objects x_i and x_j, we say x_j and x_j are associated with each other if and only if there exist a cell c, c contains both x_i and x_j, and $f(c)$ exceeds some threshold value τ.

Definition: *(union)*
c_1 and c_2 are two cells. We call cell c the union of c_1 and c_2 when both c_1 and c_2 are contained in c.

Since each object can be treated as a cell, than expression

$$Union(x_i, x_j)=Union(Cell(x_i), Cell(x_j))$$

is a generalization of the *Union* concept. From definition of the union cell, if any cell *c* contains both x_i and x_j, it contains $Union(x_i, x_j)$. From more evidence property,

$$f(Union(x_i, x_j)) \geq f(c).$$

Therefore, we can write the following equivalent data association rule: associate x_i and x_j, if $f(Union(x_i, x_j))\geq\tau$.
When the group of objects has some common characteristics and these characteristics are very different from others, given by the outlier score function, then those objects are associated [30].

3.2. Distribution-based approach

Distribution-based methods originate from statistics, where object is considered as an outlier if it deviates too much from underlying distribution. For example, in normal distribution outlier is an object whose distance from the average object is three times of the variance [18].

3.2.1. A method for high dimensional data

For high dimensional data it is better to examine the behaviour of the data in lower dimensional subspace. This is because by using full dimensional distance measures, it would be more difficult to detect outliers effectively because of the averaging behaviour of the noisy and irrelevant dimensions. So, in [2] outliers are defined by checking of those projections of the data, which have abnormally low density. Abnormally lower dimensional projections are those, in which the density of the data is exceptionally different from average density. Let the data be divided by fraction $f = 1/\phi$, where ϕ is the number of grid lines. Consider a *k*-dimensional cube, which is created by picking grid ranges from *k* different dimensions. Then, we calculate the sparsity coefficient *S(D)* of a cube *D* as follows:

$$S(D) = \frac{n(D) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}}, \tag{3.1}$$

where *n(D)* is the actual number of objects in the cell, *N* is the number of objects in the data, $N \cdot f^k$ is the expected number of objects per cell, and the divider is the standard deviation of the data objects in the cell.

The main idea behind the equation (3.1) is that coefficient accounts differences for dimensionalities of subspaces. It provides an intuitive idea that is related to the level of significance for a given projection.

Cubes that contain significantly less number of objects what expected are classified as outliers (with negative sparsity coefficient). Level of significance can be find from confidence intervals. Intervals which correspond to the first few multiples of standard deviation are illustrated in Figure 9. It means what in 99.9% the cube contains fewer objects than expected, if sparsity coefficient is negative and equal to 3 standard deviations. If we choose 2 standard deviations than can exclude useful data, for 4 standard deviations some outliers not excluded. So, we define projection as outlier if sparsity coefficient is equal to around –3.

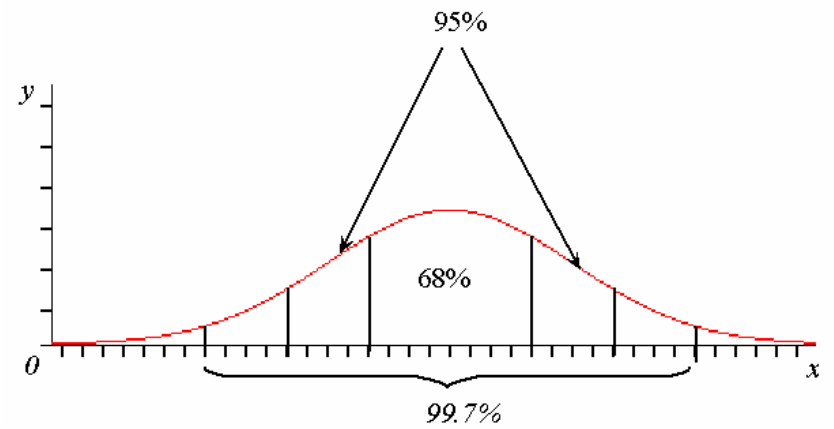

Figure 9. Normal distribution.

Consider the example for $k = 2$ and $\phi = 3$ in the Figure 10. Suppose $N = 100$, $n(D) = 2$, $N \cdot f^k = 12$, then $S(D) = -3.22$. The idea of sparsity coefficient in following: $S(D)$ is the number of standard deviations by which the actual number of objects $n(D)$ differed from the expected number of objects $N \cdot f^k$. In Figure 10 the negative sparsity coefficient defines cubes, which contain one and two objects, which are outliers.

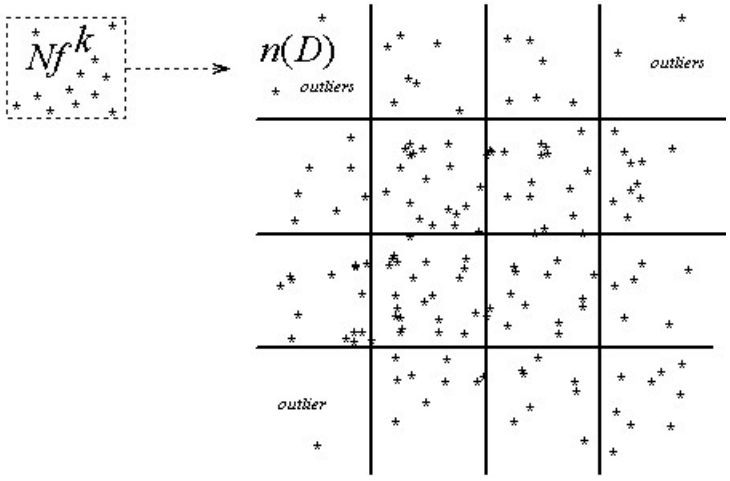

Figure 10. Outlier detection for abnormally low density.

The problem is to find the subset of dimensions, which are sparsely populated. In general, it is not possible to predict the behaviour of the data when two sets of dimensions are

combined. Therefore, the best qualitative option is to develop search methods which can identify such hidden combinations of dimensions, because of a naive brute force algorithm is very slow, and the evolutionary algorithm much faster for such aim. The idea is borrowed from a class of evolutionary search methods in order to create an efficient and effective algorithm for the outlier detection problem.

The evolutionary search technique starts with a population of random solutions and iteratively uses the process of selections, crossover and mutation in order to perform a new combination with most negative sparsity coefficient. The process continues until the population converged to a global optimum. At each stage of the algorithm, the best projections solutions were kept track of.

An important issue in the algorithm is to be able to choose the projection parameters *k* and ϕ. Values of ϕ and *k* should be picked small enough so that the sparsity coefficient of cube containing exactly one object is reasonable negative. The level of significance can be quantified by using of the normal distribution tables, because *n(D)* is assumed to fit a normal distribution. At the same time ϕ should be picked high enough so that there are sufficient number of intervals on each dimension that corresponds to a reasonable notion of locality. Let *k* be determined by using the calculation of the sparsity coefficient of an empty cube. This is given by expression (3.2) from equation (3.1)

$$s = -\sqrt{\frac{N}{\phi^k - 1}}. \tag{3.2}$$

By expressing the equation (3.2) in terms of *k* we can get the dimension of projection *k* as:

$$k = \lfloor \log_\phi (N/s^2 + 1) \rfloor.$$

The method works by finding lower dimensional projections which are locally sparse, and cannot be discovered easily by brute force techniques because of the high number of possible combinations. Such techniques for outlier detection has advantages over distance based outliers, which cannot overcome the effects of the high dimensionality [2].

3.2.2. SmartSifter algorithm

An outlier detector called *SmartSifter* (SS) is an on-line outlier detection algorithm based on unsupervised learning from data. It takes a data sequence as input in an on-line way, learns an underlying model from examples and gives a score to each object on the basis of the learned model. Thus a high score indicates a high probability that the object is an outlier. The central idea of SS is to learn the model with on-line learning algorithms and to calculate a score for a data [43].

Let $X(x, y)$ denote a dataset, where *x* denotes a vector of categorical variables, which has two or more categories, but there is no intrinsic ordering to the categories, there is no agreed way to order these from highest to lowest. The second variable *y* denotes a vector of continuous variables that can take on any value in a certain range. The joint distribution of (x, y) is $p(x, y) = p(x)p(y|x)$, where $p(x)$ is represented by a *histogram density* with a finite number of disjoint cells. A histogram density forms a probability distribution $p(x) = q_j / L_j$. Here L_j is the number of categorical variables in the j^{th} cell. q_j denotes the probability value for the j^{th} cell, so that these parameters correlate as $\sum_{j=1}^{k} q_j = 1,\ q > 0$. For each cell a

finite mixture model is used to represent the probability density over the domain of continuous variables.

A finite mixture model employs a *Gaussian mixture model*:

$$p(y|\theta) = \sum_{i=1}^{k} c_i \, p(y|\mu_i, \Lambda_i) \,,$$

where k *is* a positive integer and each *p(y)* is a *d*-dimensional *Gaussian distribution.*

So, there are as many finite mixture models as cells in the histogram density. Consider the situation where a sequence of data is given: (x_1, y_1), (x_2, y_2), … ,(x_n, y_n) in an on-line process. Identify the cell into falls given x_t and update the histogram density to obtain $p^t(x)$. Then, for that cell, update the mixture model to obtain $p^t(y|x)$. In the both cases for updating are used SDLE and SDEM algorithms, which are presented in detail in [44]. For other cells, set $p^{(t)}(y|x) = p^{(t-1)}(y|x)$. Then, SS gives a score to each value by the following equation

$$S_H(x_t, y_t) = \sum_x \int \left(\sqrt{p^{(t)}(x,y)} - \sqrt{p^{(t-1)}(x,y)} \right)^2 dy$$

This score is calculated on the basis of the models before and after updating, that is measures how much the distribution $p^{(t)}$ has moved from $p^{(t-1)}$ after learning from (x_t, y_t). The described algorithm is demonstrated as flow diagram in the Figure 11.

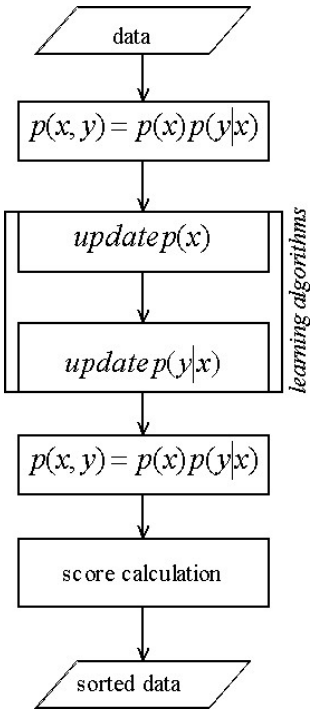

Figure 11. The flow diagram of SmartSifter algorithm.

The main advantages of method are that the computational time is inexpensive and it can deal with both categorical and continuous variables.

3.2.3. Replicator neural network approach

Replicator neural network (RNN) approach for outlier detection is a variation on the usual *regression model* where the input feature vectors are also used as the output [41]. Regression model in common case is used as an instrument for describing the dependence between input value and some factors. Thus, the RNN attempts to reproduce the input patterns in the output.

During training RNN weights are adjusted to minimize mean square error for all training patterns so that common patterns are more likely to be well reproduced by the RNN. Consequently, those patterns representing outliers are worse reproduced by the RNN and have a higher reconstruction error. The reconstruction error is used as the measure of being outlier of a given attribute as:

$$e = \frac{1}{NK}\sum_{i=1}^{N}\sum_{j=1}^{K}(x_{ij} - o_{ij})^2 \text{ , } \forall i = 1, 2, ..., N, \quad j = 1, 2, ..., K \text{ ,} \tag{3.3}$$

where N is the number of objects in the training set, K is the number of attributes. The attribute of the output from the RNN is o_{ij} [16].

For outlier detection *Outlier Factor* (OF) is defined as measure of the i^{th} data object. OF_i is the average reconstruction error over all features defined as:

$$OF_i = \frac{1}{K}\sum_{j=1}^{K}(x_{ij} - o_{ij})^2 \text{ .}$$

This is calculated for all data using the trained RNN to score each data object [41].

3.3. Density-based approach

Density-based methods have been developed for finding outliers in a spatial data. These methods can be grouped into two categories called multi-dimensional metric space-based methods and graph-based methods. In the first category, the definition of spatial neighborhood is based on Euclidean distance, while in graph-based spatial outlier detections the definition is based on graph connectivity. Whereas distribution-based methods consider just the statistical distribution of attribute values, ignoring the spatial relationships among items, density-based approach consider both attribute values and spatial relationship [38].

3.3.1. Local Outlier Factor

Local Outlier Factor (LOF) is the density-based method, which detects local outliers based on the local density of an object's neighborhood. LOF is intuitively a measure of difference in density between an object and its neighborhood objects [21]. We refer to LOF as a method from multi-dimensional metric space-based category of density-based approach. In a multidimensional dataset it is more meaningful to assign for each object a

degree of being an outlier. The key difference between LOF approach and existing notions of outliers is that being outlier is not a *binary property*.

Local outliers are the objects which relative to their local neighborhoods with respect to the densities of the neighborhoods. Formal definition of local outliers was developed in [6]:

> ***Definition**: (k-distance of an object x_i)*
> *For any positive integer k, the k-distance of object x_i, denoted as k-distance(x_i), is defined as the Euclidean distance $d(x_i, x_j)$ between x_i and an object $x_j \in X$ such that:*
> *(i) for at least k objects $x_j' \in X \setminus \{ x_i \}$ it holds that $d(x_i, x_j') \leq d(x_i, x_j)$, and*
> *(ii) for at most k-1 objects $x_j' \in X \setminus \{ x_i \}$ it holds that $d(x_i, x_j') < d(x_i, x_j)$.*

Intuitively, *k-distance(x_i)* provides a measure on the sparsity or density around the object x_i. When the *k-distance* of x_i is small, it means that the area around x_i is dense and vice versa [21].

> ***Definition:** (k-distance neighborhood of an object x_i)*
> *The k-distance of x_i, the k-distance neighborhood of x_i contains every object whose distance from x_i is not greater than the k-distance, i. e.*
>
> $$N_{k-distance(x_i)}(x_i) = \{x_l \in X \setminus \{x_i\} \mid d(x_i, x_l) \leq k - distance(x_i)\}.$$
>
> *These objects x_l are called the k-nearest neighbors of x_i.*

> ***Definition:** (reachability distance of an object x_i w.r.t. object x_j)*
> *The reachability distance of object x_i with respect to object x_j as defined as*
>
> $$reach\text{-}dist_k(x_i, x_j) = \max \{ k\text{-}distance(x_j), d(x_i, x_j) \}.$$

If object x_i is far away from x_j, then the reachability distance between the two is simply their actual distance. However, if they are close, the actual distance is replaced by the *k-distance* of x_j [6].

> ***Definition:** (local reachability density of x_i)*
> *The local reachability density of an object x_i is the inverse of the average reachability distance from the k-nearest neighbors of x_i:*
>
> $$lrd_k = 1/\left(\frac{\sum_{x_j \in N_{k-distance(x_i)}} reach-dist_k(x_i, x_j)}{\left| N_{k-distance(x_i)}(x_i) \right|} \right).$$

> ***Definition:** (local outlier factor of x_i)*
>
> $$LOF_k(x_i) = \frac{\sum_{x_j \in N_{k-distance(x_i)}} \frac{lrd_k(x_j)}{lrd_k(x_i)}}{\left| N_{k-distance}(x_i) \right|}$$

LOF is the average of the ratios of the local reachability density of x_i and those of x_i's k-nearest-neighbors. Intuitively, x_i's local outlier factor will be very high if its local reachability density is much lower than those of its neighbors [21].

The extension of LOF method is presented in [21], where one method for finding the top-n local outliers in large databases is considered. The strength of that method is it avoids computation of LOF for most objects. And provide users to find only *n* most outstanding local outliers.

3.3.2. Graph-based algorithms

Some of graph-based methods have been proposed in [18]. In the first method, named *Outlier Detection using Indegree Number* (ODIN) algorithm, outliers defined using *k-nearest neighbor* (kNN) graph. kNN graph is a weighted directed graph, in which every vertex represents a single vector, and the edges correspond to pointers to neighbor vectors. Every vertex has exactly k edges to the k nearest vectors according to a given distance function. Weight of the edge e_{ij} is the distance between vectors v_i and v_j. A simple example of kNN graph for 4 vectors with $k = 3$ is illustrated in Figure 12.

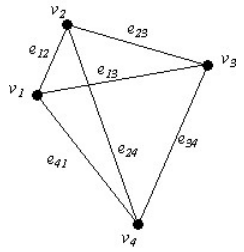

Figure 12. Illustration of kNN graph.

We define outlier detection problem for methods from this category in the following way: graph G is presented as $G = \{S, E\}$, where S is a dataset of vertex and E is a collection of edges between locations in S. The definition of outliers is given below:

> ***Definition:*** *Given kNN graph G for dataset S, outlier is a vertex, whose indegree is less than equal to T.*

In the first step of ODIN, a kNN graph is created for dataset S. Then, if vertex i has an indegree of T or less, mark it as an outlier and otherwise mark it as an inlier. The method has two control parameters: the number of outgoing edges k and the indegree threshold T.

One more method is MeanDIST algorithm, which is defined as the mean of k nearest distances. It has been modified from method proposed in [35], which calculates kNN sparseness estimate for all vectors in dataset S. But instead of sorting in an ascending order, MeanDIST algorithm cuts objects in the sorted list specified by considering differences $L_i - L_{i-1}$. Where L_i is mean distance of i^{th} vector to nearest. Then the ordered list is scanned from smaller to larger distances and calculate threshold T:

$$T = \max(L_i - L_{i-1}) * t, \tag{3.5}$$

where $t \in]0, 1[$ is a user defined parameter. In the first step of MeanDIST algorithm we compute threshold T as described above. Than we calculate kNN graph of S and sort vectors in ascending order by kNN density. If distance $L_i - L_{i-1} \geq T$, mark L_i is outlier.

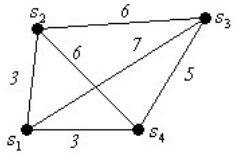

Figure 13. Example of kNN graph for KDIST algorithm.

The application example of MeanDIST algorithm is shown in Figure 13. Let be defined kNN graph for $k = 3$ and weights determined as: $e_{12} = 3$, $e_{23} = 6$, $e_{13} = 7$, $e_{41} = 3$, $e_{24} = 6$, $e_{34} = 5$. We find values of L_i as: L_1 is mean of e_{12}, e_{41} and e_{13}, L_2 is mean of e_{12}, e_{24} and e_{23}, L_3 is mean of e_{23}, e_{13} and e_{34}; and L_4 is mean of e_{41}, e_{24} and e_{34}. So we have $L_1 = 4.3$, $L_2 = 5$, $L_3 = 6$, $L_4 = 4.6$. After sorting, we construct new sequence L_1, L_4, L_2, L_3 and then consider differences $L_3 - L_2$, $L_2 - L_4$, $L_4 - L_1$. Let $t = 0.5$, in this case $T = 0.5$. We can propose what vertex s_3 is outlier, because of difference $L_3 - L_2 = 1$ exceeds T.

In [8] was proposed a *Mutual k-Nearest Neighbour* (MkNN) graph approach. It uses a special case of kNN graph: there exists an undirected edge between two vectors v_i and v_j if they belong to each others k-neighborhood. Connected components are considered as clusters, if they contain more than one vector. Isolated vectors are denoted as outliers.

Another graph-based algorithm for detecting outliers is designed in [38]. It provides a cost model for outlier detection procedure and design efficient fast algorithms to detect spatial outliers.

Consequently, in this kind of approaches, there are two parameters that define the notion of density: a minimum number of objects and a parameter specifying a volume. They determine a threshold. That is, objects are connected if their neighborhood densities exceed the given density threshold.

4. PROPOSED METHOD

The ability to detect outliers can be improved using a combined perspective from outlier detection and cluster identification. Some clustering-based algorithms like DBSCAN and ROCK [9], [12] can also handle outliers, but their main concern is clustering dataset, not outlier detection. Unlike the traditional clustering-based methods, the proposed algorithm provides much efficient outlier detection and data clustering capabilities in the presence of outliers. This approach is based on filtering of the data after clustering process. The purpose of our method is not only to produce data clustering but at the same time to find outliers from the resulting clusters.

The algorithm of our outlier detection method is divided into two stages. The first stage provides *k*-means process and in the second stage outliers are removed according to a chosen threshold. *k*-means clustering algorithm is explained in Section 2 in more detail. In this section, we will consider our proposed method and implementation of outlier removal stage. Figure 14 on left shows a more detailed view of the whole algorithm. As input parameters we need dataset *X*, number of clusters *M*, threshold *Th* and the number of iterations *R*. An initial solution is produced by the same way as for *k*-means algorithm. Next follow *k*-means clustering and outlier removal stages. After removal it is necessary to clean old partition, so in this diagram it is performed by `FreePartitioning` procedure. Then we again create optimal partitioning like in *k*-means. The stop rule is defined as the difference between the initial number of objects in the dataset, and the resultant number. It would be also possible to stop for certain number of iterations *R*, in case if the number of algorithm iterations *i* will be equal to *R*.

Pseudo code for the proposed *Clustering Outliers Removal* (COR) algorithm is listed in Figure 15. To describe it we use the same terminology as defined in Section 2. In addition we summarize data structures:

$X[N]$ array of N K-dimensional data objects
$P[N]$ array of N integer objects from X to C (partitions)
$C[M]$ array of M K-dimensional vectors cluster representatives (centroids)
P_j number of objects x_i which belong to j^{th} partition

$$P_j = \sum_{p_i = j} 1, \quad \forall\ j \in [1, M].$$

The procedure begins by creating an initial solution. The REPEAT-UNTIL loop in the next step iterates until the number of objects will not change in the dataset. In addition, it can also stop, if the number of iterations becomes *R*. Inside the loop, we perform *k*-means iterations and outlier removal stages.

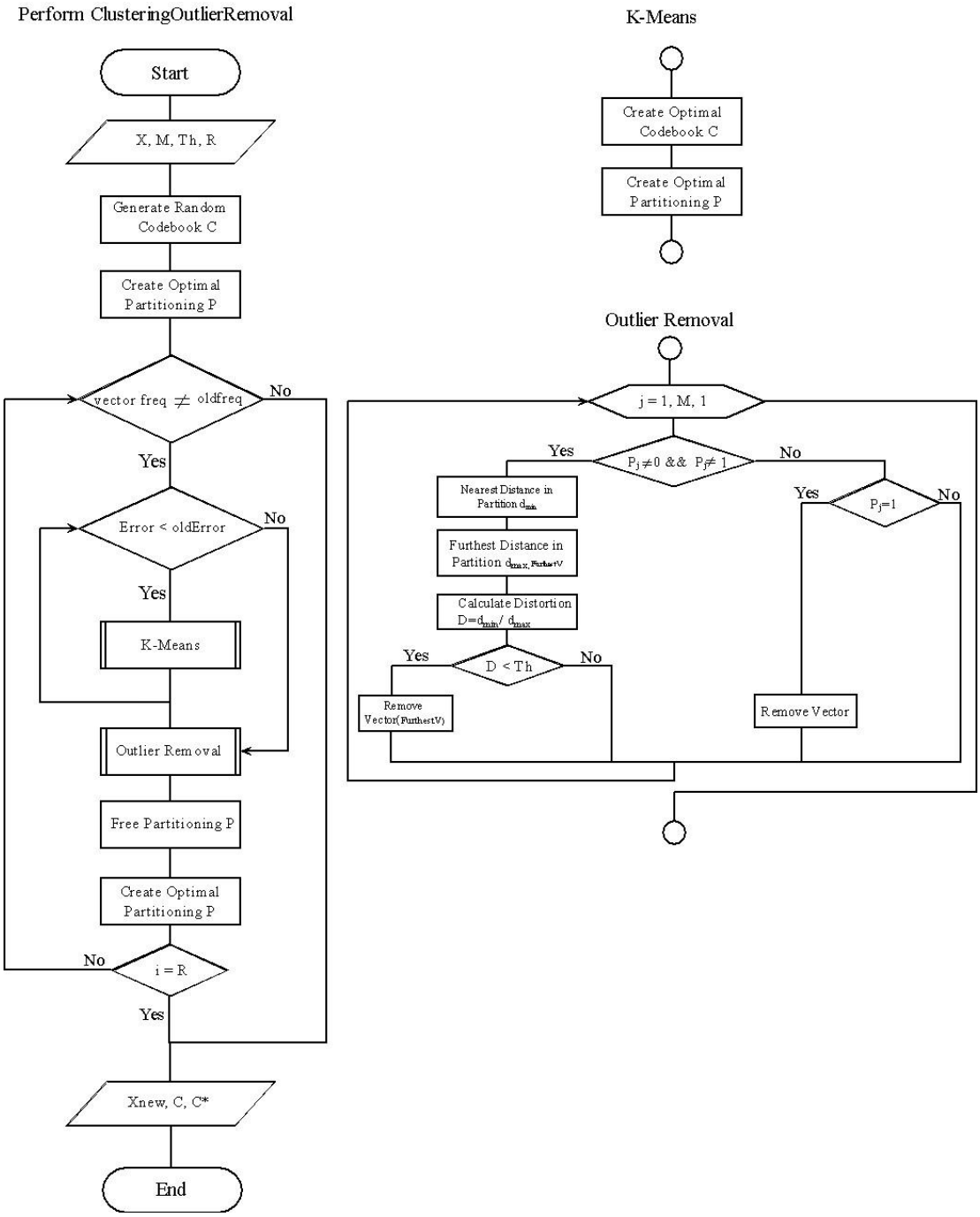

Figure 14. Flow diagram of the proposed algorithm (left). Flow diagram of *k*-mean and outlier removal functions (right).

```
PerformCOR(X)
{
 // initial solution
i := 0;               // i is a number of iterations
C := SelectRandomRepresentatives(X);
P := OptimalPartition(C, X);
REPEAT
 {
  i := i+1;
  // number of separately represented vectors in the dataset
  freq := X[N];
  // calculate average distance between each vector and its
  //current codevector
  error := AverageErrorForSolution(X, C, P);
  REPEAT
   {
    error^new:= error;
    (P^new, C^new) := K-means(P, C, X);
    error := AverageErrorForSolution(X, C^new, P^new);
   }
  UNTIL (error < error^new);
  X^new := OutlierRemoval(P^new, X, C^new);
  freq^new := X^new[N];
  FreePartitioning(P^new);   //  delete partition
  P := OptimalPartition(C^new, X^new);
  (X, C) := (X^new, C^new);
  // it would be also possible to stop for R iterations
  IF ( i = R ) BREAK;
 }
// stop when number of objects will not change in the data
UNTIL (freq != freq^new);
}
```

Figure 15. Pseudo code of the COR algorithm.

Figure 14 on right illustrates in greater detail outlier removal stage of the proposed algorithm. The principle of outliers removing depends on the difference between threshold and distortion. Threshold is set by user in range between 0 and 1. Distortion is calculated to show how far the furthest object lies from the nearest object in the partition. Under furthest and nearest object in partition we mean maximum and minimum distances correspondingly between objects and their current centroid. So, distortion is defined as the ratio of nearest distance to furthest distance in the same partition.

Removal of the most distant object occurs under the following condition: if distortion is less than threshold, the furthest object is considered to be outlier for this partition. In cases if there is just one object in partition, this object is removed too. Sometimes it happens that we have empty partitions after the clustering process, i.e. no objects are mapped to the partition. In this situation, such partition is ignored for calculating distortion. Implementation design of outlier removal stage with such kind of exceptions is explained in the pseudo code in Figure 16.

```
OutliersRemoval (P, X, C)
{
 // through all clusters
 FOR j := 1 TO M DO
  {
  // through all objects in cluster 'j'
  IF  Pj != 0 && Pj != 1 THEN {
 //find minimum distance between vectors and their current
 //codevector
   dmin := NearestDistanceInPartition(Pj, j);
   // find maximum distance between vectors and their current
   //codevector; and index of furthest vector in a partition
   (Furthest, dmax) := FurthestDistanceInPartition(Pj, j);
   distortion = dmin / dmax;        // calculate distortion
   IF  distortion < Th  THEN {
   //remove vector with index 'Furthest'
   Xnew := RemoveVector(Furthest);
   ELSE   j = j + 1;  }
  }
  ELSE  IF Pj = 1  THEN {
  //remove vector, because only one in partition
  Xnew := RemoveVector(IndexV)
     ELSE  j = j + 1;
  }
  X := Xnew;
  }
RETURN X;
}
```

Figure 16. Pseudo code of the *OutliersRemoval* function.

The procedure begins with FOR operator, in which is looking for all objects in each cluster and calculate distances to their current codevectors. After maximum and minimum distances are chosen, in the next step calculate distortion and it compare with threshold value. Then outliers are removed on the base of conditions for empty partitions and only one object in the partition.

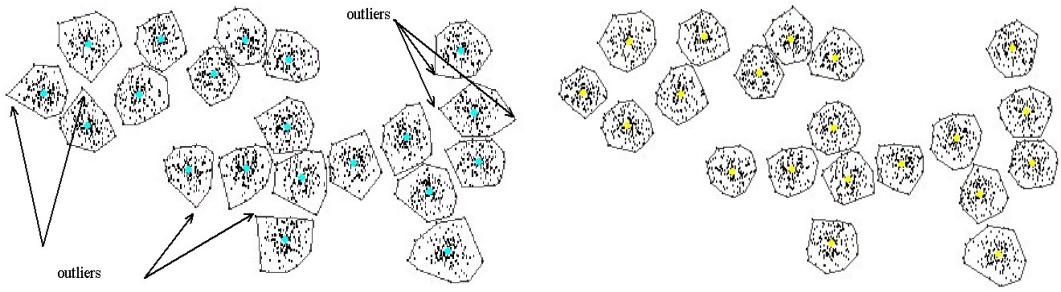

Figure 17. Original data DATA_A1 (left), and after outliers have been removed (right).

The performance of proposed algorithm for 5 iterations with threshold 0.009 is illustrated in Figure 17. In the original data, there are the most distant objects from centroids, some of such furthest objects are labeled by arrows in Figure 17 on the left. The algorithm proceeds

by removing the furthest objects from all partitions. Figure 17 on right demonstrates the resulting data after the algorithm.

So, we propose a clustering-based technique to identify outliers and simultaneously produce data clustering. Proposed outlier detection process at the same time is effective for extracting clusters and very efficient in finding outliers. But the side effect is that we actually do not know how to choose threshold for outlier removal stage.

5. EXPERIMENTAL RESULTS

In this section we present the results of an experimental study on synthetic unlabeled data and then on KDD Cup 1999 real-life labeled datasets [49] prepared for intrusion detection. The purpose of the experiment on real data was to detect intrusions.

5.1. Evaluation results

We have applied the *Receiver Operating Characteristic* (ROC) analysis to evaluate the performance of the evolved method. In each of ROC plot, the x-axis is the *False Acceptance* (FA) rate, it indicates the percentage of normal connections classified as an intrusion. FA is calculated as a number of inliers detected as outliers divided by all detections. The y-axis is the *False Rejection* (FR) rate; it indicates the percentage of not detected outliers. FR is calculated as a number of not detected outliers divided by all outliers. A data object in the down left corner of the plot with FA and FR axes corresponds to optimal performance, i.e., low FA rate with low FR rate. *Half Total Error Rate* (HTER) is a combination of FR and FA values. We will calculate HTER values and show how they are change with varying threshold and number of clusters. HTER define as (FR+FA)/2. Similar evaluation methodology has been used in [18], [28].

5.2. Experiments with synthetic data

We ran our algorithm on the synthetic *DATA_A1* dataset; it contains 3000 objects grouped in 20 clusters. *DATA_A1* is illustrated in Figure 20 on left. At first, we should discuss some formal criteria. Given a dataset *TS* = *DATA_A1*, *C* is a codebook optimized for that dataset. *TS** is a dataset *TS* from which outliers have been removed, and *C** is a codebook optimized for *TS**. *C0* are original cluster centroids of the *TS* dataset, they were used for generating *TS*. *TS**, *C*, *C** we got after testing and used theirs to evaluate the efficiency of results. Thereto we calculated error for *TS* and *C* as *f(TS, C)* it means an average error from data to cluster centroids before any removing. Those errors have not much different from each other for parameter of various threshold values. Training error *f(TS*, C*)* means an average error from resultant data to their cluster centroids, its measure means the more outliers we remove, the less training error we get. Test error *f(TS, C*)* is an average error from original dataset to the resultant cluster centroids. It shows how much the distances are changed after removing. Also were measured the differences between the original clusters centroids and centroids of clustering process *f(C0, C)*, *f(C0, C*)* is the differences between the original clusters centroids and resultant centroids. And the differences between centroids of clustering process and resultant centroids are presented as *f(C, C*)*, it the bigger, the more outliers we remove.

In these experiments we fix the number of clusters to 20 for the *DATA_A1* and the threshold value ranging between 0.0001 and 0.1. In Table 1 are shown outlier detection

results. Here the second column represents the number of removed objects. The errors described above are listed in Table 1 as well.

Table 1. Performance results for DATA_A1 dataset.

threshold	outliers	error *f(TS, C)*	training error *f(TS*, C*)*	test error *f(TS, C*)*	*f(C0, C)*	*f(C0, C*)*	*f(C, C*)*
0.0001	2	2024389	2019963	2024418	3458	3711	27
0.0002	6	2024389	2009233	2024549	3458	4153	155
0.0003	26	2024389	1955907	2025004	3781	3458	603
0.0004	46	2024376	1916140	2025317	3155	4697	896
0.0005	51	2024376	1905228	2025503	3155	5021	1070
0.0006	66	2024389	1873019	2025776	3458	4271	1299
0.0007	85	2024389	1831758	2025874	3458	4329	1387
0.0008	97	2024389	1812992	2026108	3458	4322	1578
0.0009	120	2024376	1757738	2026554	3155	4477	2039
0.0010	157	2024376	1676882	2029102	3155	6012	4527
0.0020	315	2024389	1436421	2034449	3458	10272	9544
0.0030	457	2024389	1235626	2037178	3458	13166	12250
0.0040	584	2024376	1070212	2042436	3155	18061	17425
0.0050	745	2024376	922081	2051081	3155	26982	23251
0.0060	924	2024389	763320	2060465	3458	33477	31767
0.0070	1030	2024389	663906	2069434	3458	40889	39703
0.0080	1095	2024376	605353	2072014	3155	43127	41361
0.0090	1125	2024376	574180	2074973	3155	45917	44854
0.0100	1237	2024376	513739	2084782	3155	55925	49896
0.0200	1725	2024376	244793	2104332	3155	76553	68295
0.0300	2009	2024376	145301	2145282	3155	120776	99880
0.0400	2200	2024376	95303	2197788	3155	175409	143335
0.0500	2406	2024389	57325	2204561	3458	181442	152735
0.0600	2532	2024389	27768	2224101	3458	203086	169347
0.0700	2563	2024376	25051	2221189	3155	198709	166888
0.0800	2669	2024376	13433	2252187	3155	230107	181105
0.0900	2707	2024389	10543	2258766	3458	234962	185537
0.1000	2725	2024389	9615	2266675	3458	242618	190005

From Table 1 we can conclude that *f(C0, C)* errors are less then *f(C0, C*)* errors for all the threshold values except only one, it means, original cluster centroids can be found more accurately by clustering the set *TS*. If to compare *f(TS*, C*)* and *f(TS, C*)* errors, we can observe that training error decrease as threshold value becomes bigger, but test error increase with growth of threshold. Error rates for comparison test and training errors is better visually observed in Figure 18.

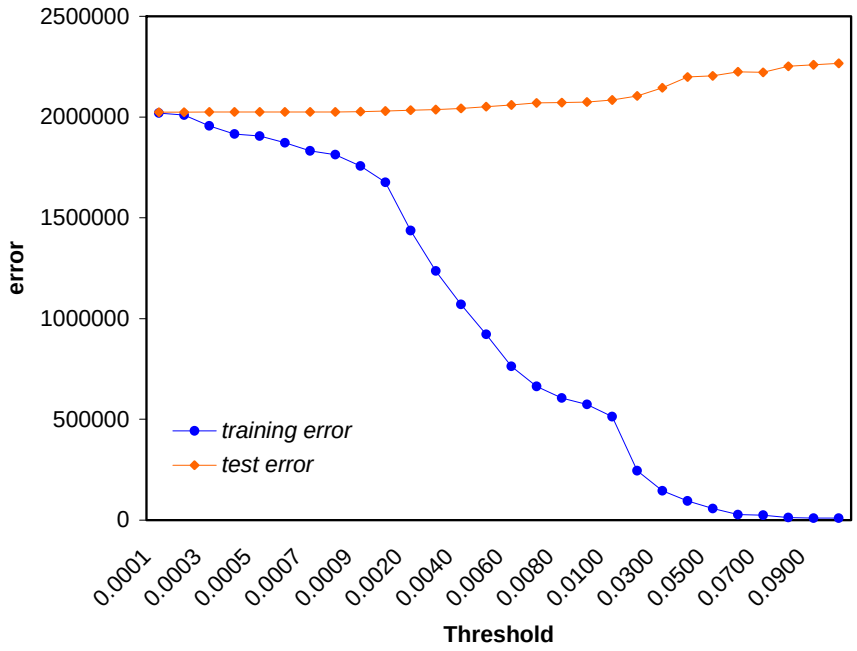

Figure 18. Comparison of training and test errors.

From Figure 18 we summarize that training error decreases very fast. It occurs as the threshold becomes bigger, thus removing more and more number of objects and the distances are decrease from dataset to the cluster centroids. The test error increases not so very fast because of cluster centroids moved at smaller distance relatively to the distance from removed outliers.

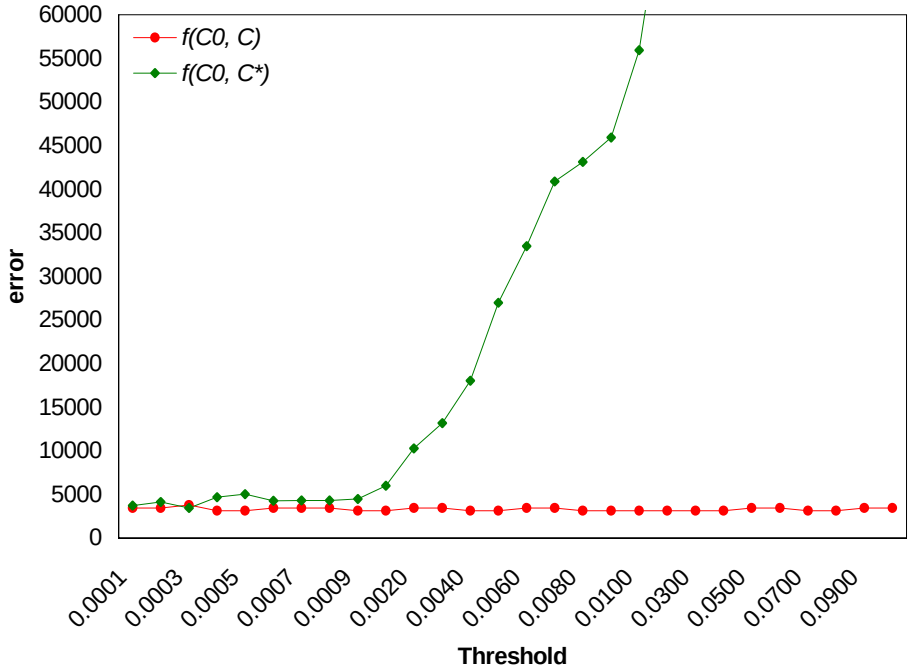

Figure 19. Comparison of *f(C0, C)* and *f(C0, C*)* errors.

The comparison of *f(C0, C)* and *f(C0, C*)* errors is summarized in

Figure 19. We observe that choosing threshold value less than 0.0009 we have insignificant difference between two error curves. They are most close to each other for threshold value ranging between 0.0001 and 0.0009. It means that resultant cluster centroids are moved not far away from original centroids on this segment, but then threshold is bigger, the *f(C0, C)* curve is sharply increase. Thus, for threshold value bigger than 0.0009 we have centroids are moved far away and probably we lose useful data for those parameters.

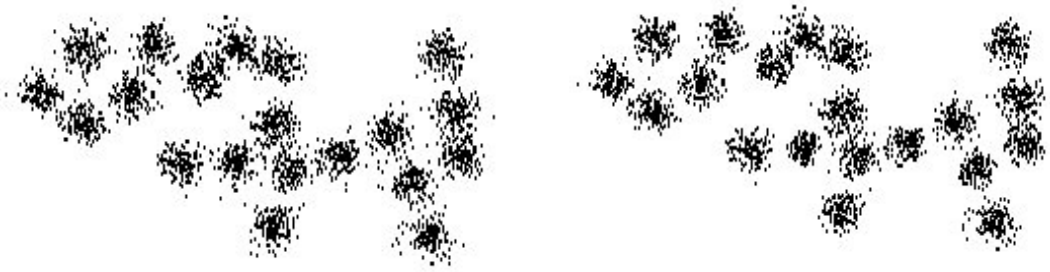

Figure 20. Original data DATA_A1 with cluster centroids (left), and the dataset after performance of the algorithm (right) for threshold value 0.0009, when 120 outliers have been removed.

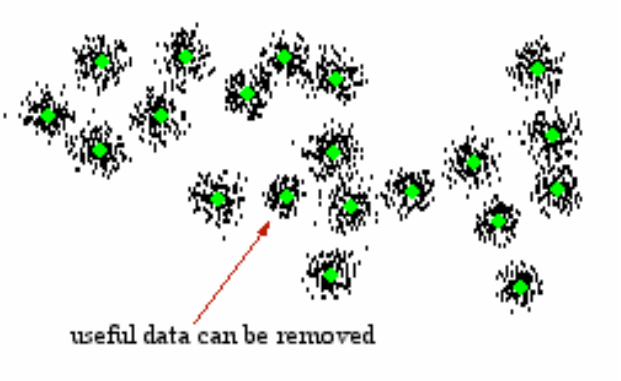

Figure 21. Dataset after performance of the algorithm for threshold value 0.001, when 157 outliers have been removed.

The second example, shown in Figure 21, was produced also from *DATA_A1* dataset with threshold value 0.001. Visually, by comparison with previous picture, one can see that if we will take bigger threshold value we, in fact, will be losing useful data, as in cluster marking by arrow, it consists only useful data. So for bigger threshold this useful data can be removed.

In general, we deduce, it was correct to choose threshold is equal to 0.0009 to get at the same time good data filtering and very well clustering of the data.

5.3. Network intrusion detection

The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The purpose was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes [49].

KDD Cup 1999 dataset is an extension of DARPA'98 dataset with a set of additionally constructed features. And it does not contain some basic information about the network connections; e.g. start time, IP addresses, ports, etc.

Dataset is specified by 41 attributes (34 continuous and 7 categorical) and a label describing the attack type (22 kinds: normal, back, buffer_overflow, ftp_write, wazermaster, etc.) where all labels except "normal" indicate in attack. Table 2 shows examples of connection records. As in [44], we used four of the original 41 attributes (service, duration, src_bytes, dst_bytes) because these four were thought of as the most basic attributes. Service is a categorical feature while the other three are continuous

features. The list and description of chosen features shown in Table 3. The range of service is {http, smtp, finger, domain_u, auth, telnet, ftp, eco_i, ntp_u, ecr_i, other, pop_3, pop_2, ftp_data, ssh, gopher, domain, private, login, imap4, time, shell, IRC, urh_i, X11, urp_i, tftp_u, discard, tim_i, red_i, nntp, uucp,netbios_ssn, daytime, echo}. The number of service kids is 35, and we divided them into five subsets according to the five feature: *http*, *smtp*, *ftp*, *ftp_data*, *others*; because each categorical variable belonging to "others" has a low frequency.

Table 2. Network connection records.

Duration	protocol_type	service	flag	src_bytes	dst_bytes	land	…	label
0	tcp	http	SF	219	1337	0	…	normal
0	tcp	http	SF	217	2032	0	…	normal
0	icmp	ecr_i	SF	1032	0	0	…	smurf
9400	udp	other	SF	147	105	0	…	normal
13	tcp	smtp	SF	11994	1361	0	…	normal
…	…	…	…	…	…	…	…	…
25	tcp	ftp	SF	334	1063	0	…	normal
0	tcp	Private	S0	0	0	0	…	neptune

Table 3. List of features.

Feature name	Description	Type
service	network service on the destination, e.g., http, telnet, etc.	discrete
duration	length (number of seconds) of the connection	continuous
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous

The original dataset contains 4,898,431 data, including 3,925,651 attacks (80.1%). This rate of attacks is too large. Therefore we took a sub dataset, which produced by picking up the original 10 % KDD Cup 1999 dataset. Details of the resultant dataset are listed in Table 4. Then we normalized it, where each continuous attribute values were concentrated around 0 according to the following equation:

$$y=log(x+0.1).$$

Table 4. The extracted datasets from KDD Cup 1999.

Service	Events	Intrusions	Proportion
http	43047	74	0.17%
smtp	9187	4	0.04%
ftp_data	307	78	25.41%
ftp	619	266	42.97%
others	3908	260	6.65%

5.4. Experiments with real data

Experiments were run on *ftp*, *ftp_data*, *smtp*, *others* and *http* datasets, that were described in previous section. The aim is to identify intrusions within each of the categories by identifying outliers. Figure 22, Figure 23, Figure 24, Figure 25 and Figure 26 illustrate original datasets extracted from KDD Cup 1999, on left and the results after running algorithm are illustrated on right. Here, by stars are shown cluster centroids for resultant datasets.

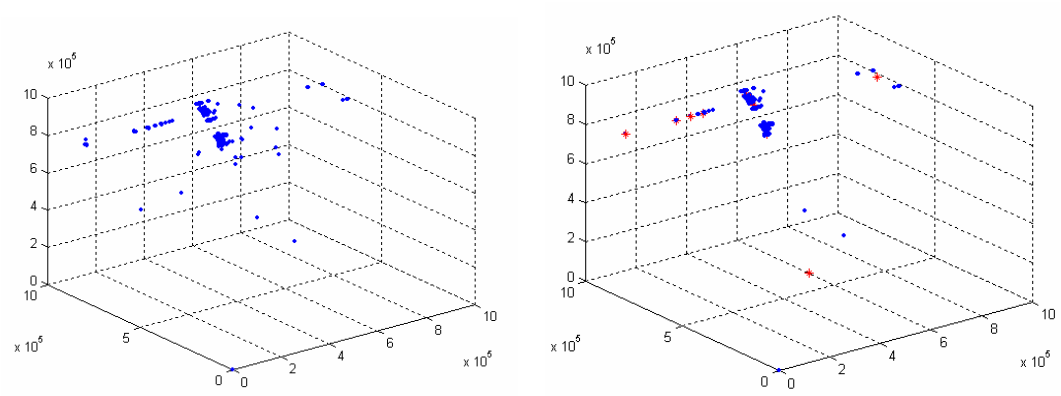

Figure 22. Visualization of *ftp* dataset before outlier removing (left). Resultant *ftp* after applying proposed method with threshold value 0.005 and number of clusters 10 (right).

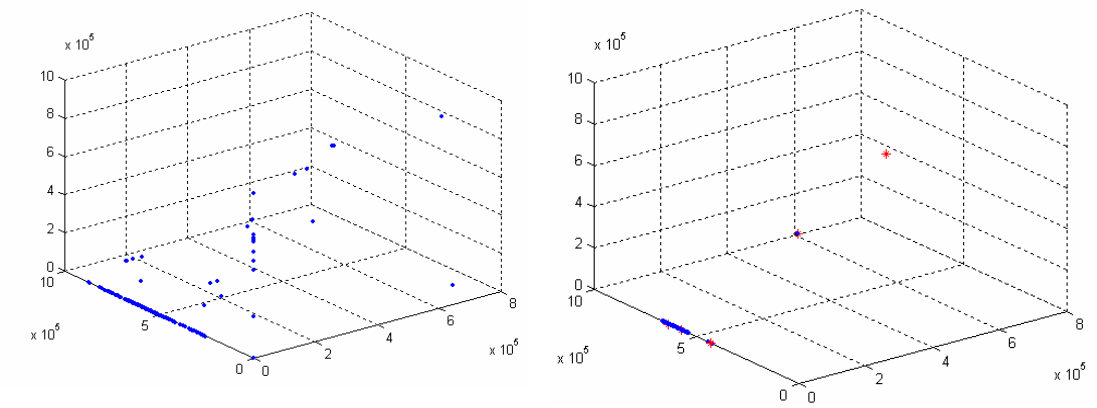

Figure 23. Visualization of *ftp_data* dataset before outlier removing (left). Resultant *ftp_data* after applying proposed method with threshold value 0.9 and number of clusters 5, is performed with 35 iterations (right).

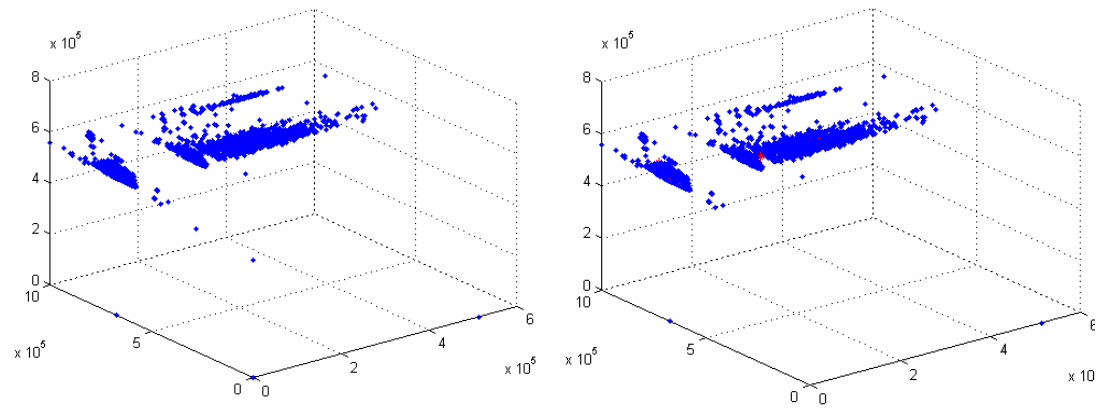

Figure 24. Visualization of *smtp* dataset before outlier removing (left). Resultant *smtp* after applying proposed method with threshold value 0.0000005 and number of clusters 5 (right).

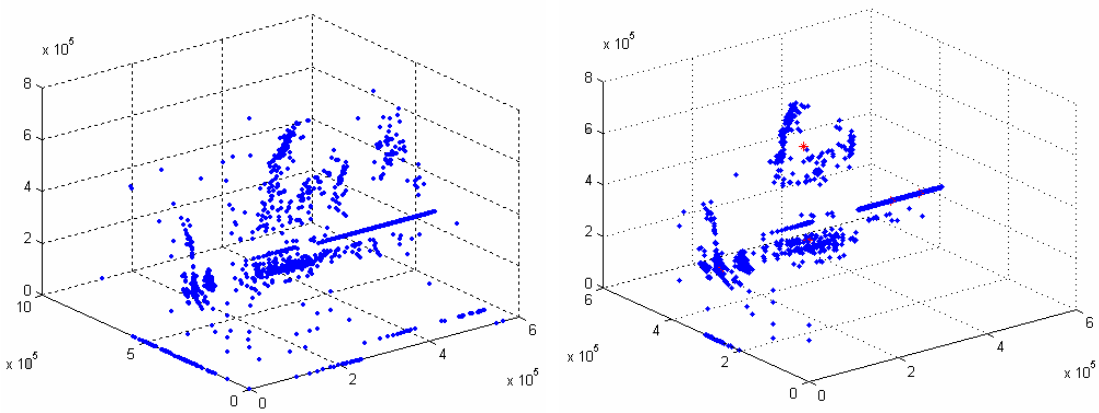

Figure 25. Visualization of *others* dataset before outlier removing (left). Resultant *others* after applying proposed method with threshold value 0.05 and number of clusters 5 (right).

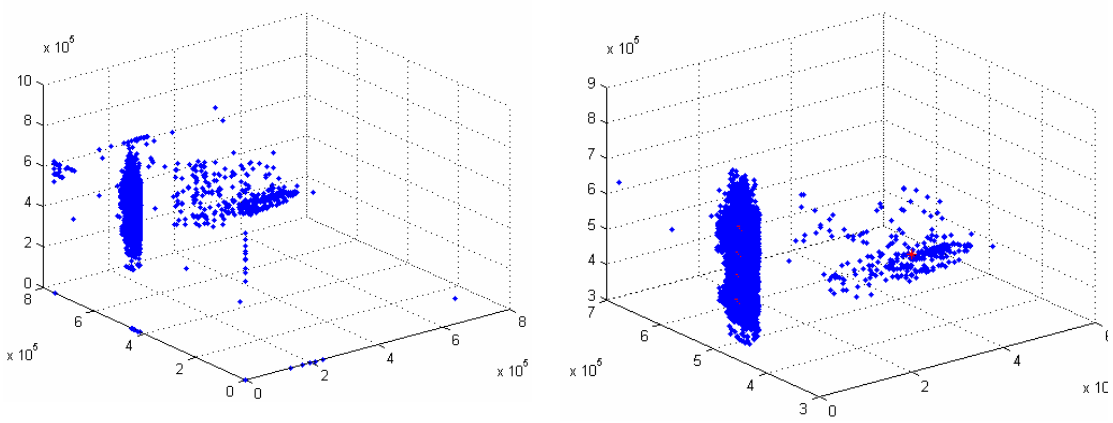

Figure 26. Visualization of *http* dataset before outlier removing (left). Resultant *http* after applying proposed method with threshold value 0.005 and number of clusters 5, is performed with 45 iterations (right).

In these pictures, the clusters have different sizes and different shapes, and the nose has different intensities. The results demonstrate that the COR algorithm can achieve good identifying and eliminating noise.

To demonstrate the effectiveness of this approach we report a computational results of evaluation method described in Section 5.1. Table 5, Table 6, Table 7, Table 8 and Table 9 provides FA and FR rates for all five datasets enumerated above. Experiments are performed for different number of clusters and various threshold values. By marked cells in tables are represented the best minimum FA and FR errors for each of datasets.

Table 5. List of FA rate and FR rate for *ftp* dataset.

number of clusters	error rates	threshold						
		0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	FA	67%	67%	51%	38%	30%	20%	33%
	FR	66%	72%	76%	77%	89%	89%	99%
10	FA	50%	42%	28%	9%	4%	9%	33%
	FR	19%	22%	78%	36%	68%	86%	99%
15	FA	46%	43%	22%	15%	6%	14%	21%
	FR	20%	25%	58%	43%	79%	89%	95%
20	FA	50%	36%	21%	23%	27%	15%	18%
	FR	25%	30%	49%	68%	90%	87%	96%
25	FA	48%	38%	17%	28%	12%	21%	33%
	FR	32%	36%	56%	77%	83%	94%	98%
50	FA	53%	43%	40%	32%	23%	26%	25%
	FR	46%	54%	75%	77%	91%	95%	97%

Table 6. List of FA and FR rates for *ftp_data* dataset.

number of clusters	error rates	threshold						
		0.9	0.5	0.1	0.05	0.01	0.005	0.001
5	FA	71%	78%	83%	84%	90%	93%	88%
	FR	3%	24%	46%	50%	78%	85%	87%
10	FA	74%	78%	82%	79%	85%	89%	81%
	FR	3%	24%	46%	39%	79%	85%	91%
15	FA	73%	78%	80%	77%	83%	90%	88%
	FR	3%	26%	42%	41%	79%	89%	96%
20	FA	72%	77%	78%	79%	88%	94%	91%
	FR	5%	28%	43%	48%	85%	94%	96%
25	FA	72%	78%	77%	75%	87%	93%	96%
	FR	7%	30%	46%	47%	87%	93%	98%
50	FA	67%	73%	74%	69%	72%	77%	82%
	FR	8%	32%	50%	53%	79%	88%	96%

Table 7. List of FA and FR rates for smtp dataset.

number of clusters	error rates	threshold						
		0.0001	0.00005	0.00001	0.000005	0.000001	0.0000005	0.0000001
3	FA	97%	97%	94%	50%	66%	50%	33%
	FR	25%	25%	25%	50%	25%	50%	50%
4	FA	97%	97%	93%	80%	66%	40%	50%
	FR	25%	25%	25%	25%	25%	25%	50%
5	FA	97%	97%	92%	80%	57%	25%	50%
	FR	25%	25%	25%	25%	25%	25%	75%
10	FA	98%	93%	100%	88%	57%	40%	50%
	FR	25%	75%	100%	25%	25%	25%	75%
15	FA	99%	100%	87%	80%	100%	80%	100%
	FR	75%	100%	75%	50%	100%	75%	100%

Table 8. List of FA and FR rates for *others* dataset.

number of clusters	error rates	threshold						
		0.5	0.1	0.05	0.01	0.005	0.001	0.0005
5	FA	64%	70%	62%	70%	52%	87%	91%
	FR	22%	38%	21%	63%	63%	95%	96%
10	FA	83%	84%	85%	92%	92%	98%	98%
	FR	33%	45%	53%	86%	86%	97%	97%
15	FA	85%	88%	89%	94%	91%	95%	97%
	FR	23%	40%	55%	80%	71%	88%	95%
20	FA	87%	89%	93%	95%	94%	95%	97%
	FR	22%	36%	62%	76%	77%	84%	95%
25	FA	91%	88%	94%	95%	95%	96%	95%
	FR	27%	25%	60%	74%	78%	88%	93%
50	FA	94%	95%	95%	95%	94%	90%	90%
	FR	21%	45%	60%	79%	82%	88%	91%

Table 9. List of FA and FR rates for *http* dataset.

number of clusters	error rates	threshold						
		0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	FA	85%	85%	85%	85%	85%	82%	80%
	FR	1%	1%	1%	1%	1%	1%	5%
10	FA	98%	98%	92%	98%	93%	94%	91%
	FR	79%	86%	0%	86%	20%	27%	28%
15	FA	95%	95%	95%	95%	99%	99%	93%
	FR	0%	0%	0%	0%	89%	89%	21%
20	FA	96%	99%	99%	99%	99%	99%	98%
	FR	0%	85%	81%	33%	86%	95%	85%
25	FA	96%	99%	99%	99%	99%	97%	99%
	FR	0%	85	83%	86%	89%	31%	93%
50	FA	99%	99%	99%	99%	99%	99%	99%
	FR	85%	89%	87%	95%	89%	93%	95%

Analyzing the tables above we summarize that *ftp_data* and *http* datasets have very high FA rates. We can improve the outlier detection results for them by control the number of iterations. That implies that in this case the method is able to obtain the better results by decreasing the number of iterations. Table 10 and Table 11 shows FA and FR rates produced for 35 iterations on *ftp_data* dataset and 45 iterations on *http* dataset.

Table 10. List of FA and FR rates for *ftp_data* dataset, 35 iterations.

number of clusters	error rates	threshold						
		0.9	0.5	0.1	0.05	0.01	0.005	0.001
5	FA	57%	63%	67%	70%	84%	86%	77%
	FR	3%	25%	47%	50%	76%	85%	87%
10	FA	73%	77%	80%	81%	87%	90%	89%
	FR	2%	24%	35%	50%	78%	87%	94%

Table 11. List of FA and FR rates for *http* dataset, 45 iterations.

number of clusters	error rates	threshold						
		0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	FA	76%	78%	78%	75%	78%	73%	71%
	FR	2%	12%	10%	1%	29%	10%	9%
10	FA	87%	87%	89%	97%	87%	90%	98%
	FR	0%	0%	22%	81%	22%	87%	86%

From Table 10 and Table 11 we conclude that algorithm with less number of iterations on *ftp_data* and *http* datasets outperform the experiments results before iteration control. 35 iterations for *ftp_data* give minimum FA rate 57% and FR rate 3% instead of before they are were 71% and 3% correspondingly. 45 iterations for *http* dataset give minimum FA rate 75% and FR rate 1% instead before they are were 82% and 1% correspondingly.

HTER values for *ftp*, *ftp_data*, *smtp*, *others* and *http* datasets are summarized in Table 12, Table 13, Table 14, Table 15 and Table 16. By marked cells are shown minimum HTER values.

Table 12. HTER for *ftp* dataset.

number of clusters	threshold						
	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	67.10%	70.05%	63.73%	57.72%	60.30%	55.21%	66.29%
10	34.93%	32.34%	53.66%	23.01%	36.69%	47.92%	66.29%
15	33.38%	34.50%	40.59%	29.66%	43.05%	51.90%	58.64%
20	37.78%	33.51%	35.15%	46.32%	59.00%	51.87%	57.39%
25	40.59%	37.76%	37.31%	52.93%	47.72%	57.70%	65.91%
50	49.83%	48.82%	58.06%	55.38%	57.34%	61.26%	61.37%

Table 13. List of HTER for *ftp_data* dataset.

number of clusters	threshold						
	0.9	0.5	0.1	0.05	0.01	0.005	0.001
5	37.71 %	51.21%	64.93%	67.04%	84.27%	89.69%	87.84%
10	38.94%	51.33%	64.32%	59.65%	82.33%	87.85%	86.30%
15	38.57%	52.58%	61.53%	59.50%	81.58%	90.05%	92.52%
20	39.01%	53.07%	61.06%	63.99%	87.09%	94.49%	93.66%
25	40.05%	54.40%	61.96%	61.58%	87.49%	93.63%	97.69%
50	38.42%	52.36%	62.41%	61.79%	75.95%	82.98%	89.25%

Table 14. List of HTER for *smtp* dataset.

number of clusters	threshold						
	0.0001	0.00005	0.00001	0.000005	0.000001	0.0000005	0.0000001
3	61.01%	61.02%	59.86%	50.00%	45.83%	50.00%	41.66%
4	61.42%	61.35%	59.30%	52.50%	45.83%	32.50%	50.00%
5	61.47%	61.35%	58.55%	52.50%	41.07%	25%	62.50%
10	61.86%	84.37%	100.00%	56.73%	41.07%	32.50%	62.50%
15	87.27%	100.00%	81.25%	65.00%	100.00%	77.50%	100.00%

Table 15. List of HTER for *others* dataset.

number of clusters	threshold						
	0.5	0.1	0.05	0.01	0.005	0.001	0.0005
5	43.64%	54.15%	41.84%	66.94%	58.06%	91.36%	93.94%
10	58.25%	65.14%	69.50%	89.87%	89.16%	97.88%	97.87%
15	54.80%	64.40%	72.68%	87.41%	81.27%	92.08%	96.38%
20	54.96%	63.21%	78.19%	85.86%	86.31%	89.93%	96.64%
25	59.46%	57.04%	77.21%	85.03%	87.39%	92.25%	94.67%
50	57.96%	70.41%	78.29%	87.66%	88.42%	89.61%	90.96%

Table 16. List of HTER for *http* dataset.

number of clusters	threshold						
	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	43.39%	43.39%	43.39%	43.39%	43.39%	41.83%	43.03%
10	89.03%	92.68%	46.30%	92.67%	56.87%	60.52%	60.04%
15	47.53%	47.53%	47.53%	47.53%	94.28%	94.28%	57.58%
20	48.15%	92.26%	90.17%	65.60%	92.94%	97.86%	91.96%
25	48.46%	92.33%	91.64%	93.02%	94.39%	64.26%	96.38%
50	92.44%	94.50%	93.82%	97.93%	94.48%	96.51%	97.82%

HTER values for *ftp_data* and *http* datasets with smaller number of iterations is represented in Table 17 and Table 18.

Table 17. List of HTER for *ftp_data* dataset, 35 iterations.

number of clusters	threshold						
	0.9	0.5	0.1	0.05	0.01	0.005	0.001
5	30.61%	44.58%	57.57%	60.44%	80.49%	86.32%	82.22%
10	37.85%	51.08%	57.98%	65.62%	82.71%	88.68%	92.30%

Table 18. List of HTER for *http* dataset, 45 iterations.

number of clusters	threshold						
	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
5	39.39%	45.28%	44.44%	38.54%	54.16%	42.36%	40.53%
10	43.84%	44.75%	43.83%	43.84%	56.23%	89.15%	55.15%

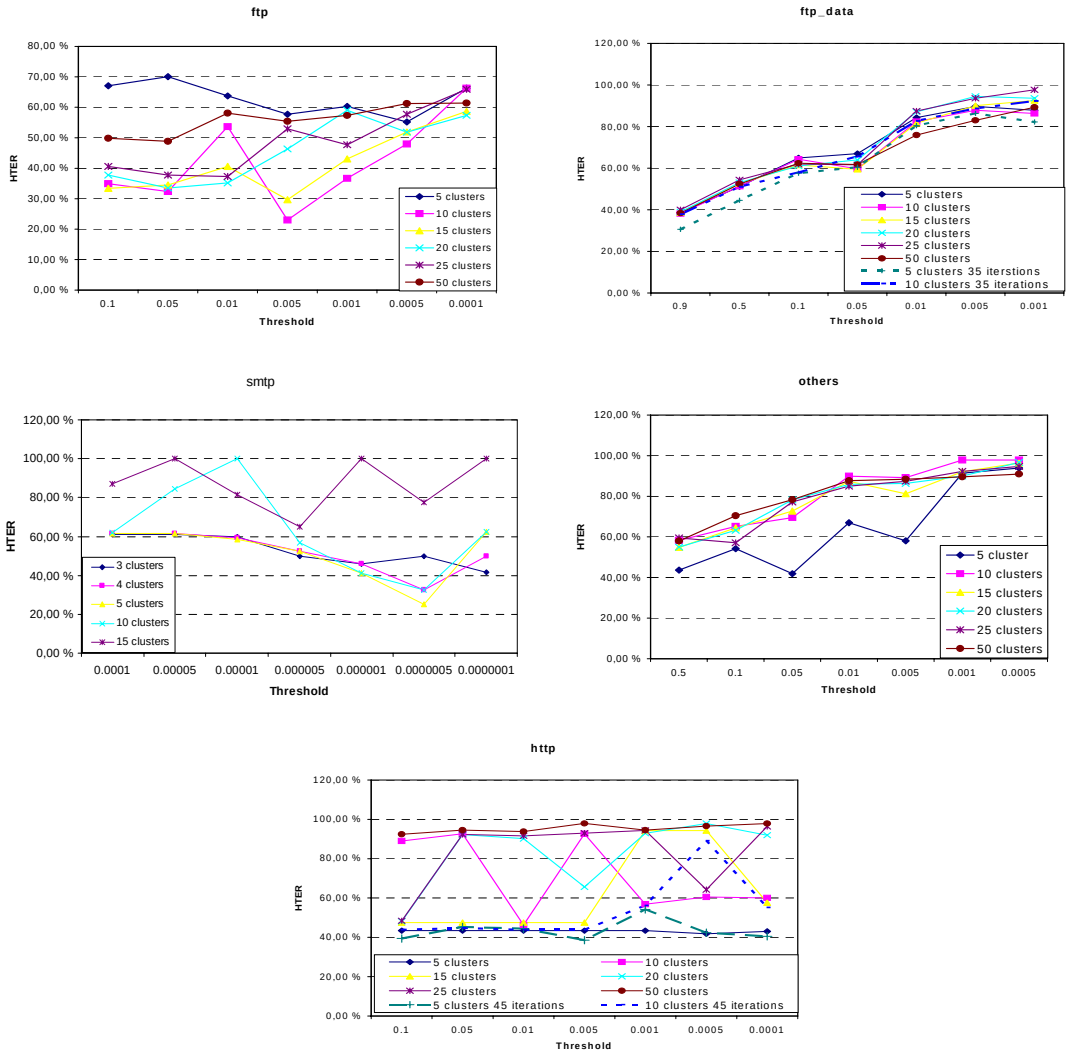

Figure 27. Error rate as a function of threshold value for tested datasets.

Figure 27 visualize the error rates as a function of threshold value for all five subdatasets. Here is shown that the best parameter values for *ftp* dataset are threshold 0.005 and the number of clusters is equal to 5. That is obvious from diagram as a lowermost object. From Table 5, we will find the best minimum FR and FA rates, with the ratios 10% and 37% correspondingly for chosen parameters. Table 12 shows that HTER is equal to 23.01% in this case.

According to the results for *ftp_data* dataset, in Table 13, we notice that minimum HTER ratio is reached 37.71%. However, if to decrease the number of iterations we will have significantly better result 30.61%, as shown in Table 17. From Figure 27 can be easily seen that by stroke lines is illustrated the minimum HTER error for threshold value 0.9 and the number of clusters is equal to 5 with 35 iterations. Table 10 shows that minimum FR and FA rates are 57% and 3% correspondingly, but before iterations decreasing the minimum FR and FA rates were 71% and 3% correspondingly, that is shown in Table 6.

An example for *smtp* dataset also is shown in Figure 27. Here is the best HTER ratio is reached 25%, it gives by threshold value 0.0000005 and number of clusters 5. Minimum FR and FA rates for each 25%, it can be find from Table 7.

Experimental results on *others* dataset are shown in Table 15. The best HTER value gives 41.84 % by threshold value 0.05 and by 5 numbers of clusters. From Table 8 can be find that FA ratio reached at 62% and FR ratio reached at 21%. Visual example of HTER curves is shown in Figure 27.

The minimum HTER value of *http* dataset can be found from Table 16, it is equal to 41.83%. If one look to Table 9 to FA and FR rates for same parameters, here can be found that FA is equal to 82% and FR is equal to 1%. We had tried to decrease such high FA rate by decreasing the number of iterations. We have tested proposed method on *http* dataset with 45 iterations and so, resultant dataset had 75% FA that is better ratio and FR is equal to 1%, that is the same. The HTER ratio in this case is reached 38.54%. An examples of how HTER curves changes with different threshold is shown in Figure 27, here is the best parameters are threshold value 0.005 and the number of clusters is equal to 5.

5.5. Comparison with other outlier detection methods

We compared the performance of ODIN and MkNN algorithms on *ftp*, *ftp_data*, *smtp*, *others* and *http* datasets against the proposed method. Table 19 shows the minimum values of HTER for all compared methods and datasets.

Table 19. Summary of the results as error rate.

Dataset	HTER		
	proposed method	ODIN	MkNN
ftp	23.01%	28.5%	59.62%
ftp_data	30.61%	37.5%	65.36%
smtp	25%	49.5%	74.61%
others	41.84%	46.5%	51.38%
http	38.54%	49.5%	65.31%

Experimentally, COR algorithm is shown to perform very well on *ftp*, *smtp* and *http* datasets. Here it significantly superior to ODIN and MkNN. It occurs due to the spherical shape of data, in this case the clustering performed much better by *k*-means algorithm. So, outlier detection is better achieved. In another experiments, the datasets have clusters with non spherical shapes. For example, on *others* dataset proposed method has not very good, but superior result to ODIN and MkNN, the HTER ratio here is reached 41.84%. While ODIN outperform COR with the *ftp_data* dataset, the result can be improved by reducing the number of iteration, for 35 iterations algorithm detect outliers with HTER ratio of 30.61%. Proposed algorithm on *http* dataset can detect intrusions with the HTER ratio of 38.54% for 45 iterations, it significantly outperforms ODIN and MkNN algorithms.

Moreover, another important observation from experimental results, that proposed method outperform the ODIN and MkNN in detecting false intrusions; it gives lower FA rates for all datasets.

According to the experiments ROC works effectively especially where the data has spherical shape. Its performance appears to degrade with datasets containing radial dataset and so it is not recommended for this type of dataset. Our study indicates that for datasets which have non spherical shape, we can improve the outlier detection results by setting the number of iterations.

In summary, the above experimental results on *ftp*, *ftp_data*, *smtp*, *others* and *http* datasets show that the proposed algorithm can identify outliers more successfully than existing algorithms.

6. CONCLUSIONS

This thesis proposes and analyzes a new outlier detection method called COR algorithm. It provides efficient outlier detection and data clustering capabilities in the presence of outliers. This approach is based on filtering of the data after clustering process. It makes those two problems solvable for less time, using the same process and functionality for both clustering and outlier identification. Moreover, we discussed the different categories in which outlier detection algorithms can be classified, i.e. density-based, distribution-based and distance-based methods.

Furthermore, we applied algorithm to a real dataset KDD Cup 1999. Experimentally, COR is shown to perform very well on several real datasets. The results indicate that COR works effectively especially where the data has spherical shape. Its performance appears to degrade with datasets containing radial shape clusters and it is not recommended for this type of datasets. This study indicates that for datasets which have non spherical shape, we can improve the outlier detection results by setting the number of iterations. The experimental results demonstrate that the proposed method is significantly better than ODIN and MkNN in finding outliers.

With simple modifications, the method can be implemented for other distance metrics. An important direction for further study is how to apply the COR algorithm to the more general case, where the number of clusters and the threshold value must also be solved. Also we can control the number of iterations. Moreover, a possible extension of this method would be to compare the performance of our method using different data clustering approaches.

The main contribution of the present work is the design of an outlier detection process. Performed experiments demonstrate that COR algorithm was successful in detecting intrusions.

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Volume 27, Issue 2, pages 94 – 105, June 1998.

[2] C. Aggarwal and P. Yu, "Outlier Detection for High Dimensional Data". In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Volume 30, Issue 2, pages 37 – 46, May 2001.

[3] C. Aggarwal and P. Yu, "Redefining Clustering for High-Dimensional Applications". In *Proceedings of the IEEE International Conference on Transaction of Knowledge and Data Engineering*, Volume 14, Issue 2, pages 210 – 225, April 2002.

[4] S. Alfuraih, N. Sui and D. McLeod, "Using Trusted Email to Prevent Credit Card Frauds in Multimedia Products". *World Wide Web: Internet and Web Information Systems*, Volume 5, Issue 3, pages 244 – 256, 2002.

[5] B. Borah, D. K. Bhattacharyya, "An Improved Sampling-based DBSCAN for Large Spatial Databases". In *Proceedings of the International Conference on Intelligent Sensing and Information*, page 92, 2004.

[6] M. Breunig, H.-P. Kriegel, R. Ng and J. Sander, "LOF: Identifying Density-Based Local Outliers". In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93 – 104, 2000.

[7] Brian S. Everitt, *"Cluster analysis"*. Third Edition, 1993.

[8] M. Brito, E. Chávez, A.Quiroz and J. Yukich, "Connectivity of the Mutual k-Nearest-Neigbor Graph in Clustering and Outlier Detection". *Statistics & probability Letters*, Volume 35, Issue 1, pages 33-42, August 1997.

[9] 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 the Second International Conference on Knowledge Discovery and Data Mining*, pages 226 – 231, 1996.

[10] P. Fränti and J. Kivijärvi, "Randomised Local Search Algorithm for the Clustering Problem". *Pattern Analysis and Applications*, Volume 3, Issue 4, pages 358 – 369, 2000.

[11] S. Giha, R. Rasstogi and K. Shim, "CURE: an efficient clustering algorithm for large databases". In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 73 – 84, June 1998.

[12] S. Guha, R. Rastogi and K. Shim, “ROCK: A Robust Clustering Algorithm for Categorical Attributes”. In *Proceedings of the 15th International Conference on Data Engineering*, page 512, March 1999.

[13] M. Halkidi, Y. Batiskakis and M. Vazirgiannis, “Clustering algorithm and validity measures”. In *Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management*, pages 3 – 22, Fairfax, Virginia, USA, July, 2001.

[14] M. Halkidi, Y. Batistakis and M. Vazirgiannis, “Cluster Validity Methods: part I”. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Volume 31, Issue 2, pages 40 – 45, June 2002

[15] M. Halkidi, Y. Batistakis and M. Vazirgiannis, “Clustering Validity Checking Methods: Part II”. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Volume 31, Issue 3, pages19 – 27, September 2002.

[16] S. Hawkins, H. He, G. Williams and R. Baxter, “Outlier Detection Using Replicator Neural Networks”. In *Proceedings of the Fourth International Conference on Data Warehousing and Knowledge Discovery*, pages 170 – 180, 2002.

[17] Z. He, X. Xu and S. Deng, “Discovering Cluster-based Local Outliers”. *Pattern Recognition Letters*, Volume 24, Issue 9-10, pages 1641 – 1650, June 2003.

[18] V. Hautamäki, I. Kärkkäinen and P. Fränti, “Outlier Detection Using k-Nearest Neighbor Graph”. In *Proceedings of the International Conference on Pattern Recognition*, Volume 3 pages 430 – 433, Cambridge, UK, August 2004.

[19] T. Hu and S. Y. Sung, “Detecting pattern-based outliers”. *Pattern Recognition Letters*, Volume 24, Issue 16, pages 3059 – 3068, December 2003.

[20] M. Jaing, S. Tseng and C. Su, “Two-phase Clustering Process for Outlier Detection”. *Pattern Recognition Letters*, Volume 22, Issue 6 – 7, pages 691 – 700, May 2001.

[21] W. Jin, A. Tung and J. Han, “Mining Top-n Local Outliers in Large Databases”. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 293 – 298, 2001.

[22] H. Jin, M.-L. Wong and K.-S. Leung, “Scalable Model-based Clustering by Working on Data Summaries”. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 91 – 98, November 2003.

[23] L. Kaufman and P. Rousseeuw, “*Finding Groups in Data: An Introduction to Cluster Analysis”*. John Wiley Sons, New York, USA, 1990.

[24] E. Knorr and R. Ng, “A Unified Approach for Mining Outliers”. In *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research*, page 11, 1997.

[25] E. Knorr and R. Ng, “A Unified Notion of Outliers: Properties and Computation”. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 219 – 222, August 1997.

[26] E. Knorr, R. Ng and R. Zamar, “Robust Space Transformation for Distance-based Operations”. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126 – 135, August 2001.

[27] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava, V. Kumar, “A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection”. In *Proceedings of the Third SIAM Conference on Data Mining*, May 2003.

[28] W. Lee, S. Stolfo, K. Mok, “A Data Mining Framework for Building Intrusion Detection Models”. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages120 – 132, May 1999.

[29] S. Lin and D. Brown, “An Outlier-based Data Association Method”. In *Proceedings of the SIAM International Conference on Data Mining*, San Francisco, CA, May 2003.

[30] S. Lin and D. Brown, “An Outlier-based Data Association Method for Linking Criminal Incidents”. In *Proceedings of the SIAM International Conference on Data Mining*, October 2004.

[31] J. Liu and P. Gader, “Outlier Rejection with MLPs and Variants of RBF Networks”. In *Proceedings of the 15th Iinternational Conference on Pattern Recognition,* Volume 2, page 680 – 683, Barcelona, Spain, September 2000.

[32] J. Liu and P. Gader, “Neural Networks with Enhanced Outlier Rejection Ability for Off-line Handwritten Word Recognition”. *The journal of the Pattern Recognition society*, Volume 35, Issue 10, pages 2061 – 2071, October 2002.

[33] E. Paquet, “Exploring anthropometric data through cluster analysis”. *Published in Digital Human Modeling for Design and Engineering (DHM)*, pages, Rochester, MI, June, 2004.

[34] M. Petrovskiy, “Outlier Detection Algorithms in Data Mining Systems”. *Programming and Computing Software*, Volume 29, Issue 4, pages 228 – 237, July 2003.

[35] C. Phua, D. Alahakoon and V. Lee, “Minority Report in Fraud Detection: Classification of Skewed Data”. *Special Issue on Learning from Imbalanced Datasets*, Volume 6, Issue 1, pages 50 – 59, 2004.

[36] S. Ramaswamy, R. Rastogi and K. Shim, “Efficient Algorithms for Mining Outliers from Large Data Sets”. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Volume 29, Issue 2, pages 427 – 438, May 2000.

[37] M. Sato, Y. Sato and L.C. Jain, "General Fuzzy Clustering Model and Neural Networks". In *Proceedings of the Electronic Technology Directions to the Year 2000*, pages 104 – 112, May 1995.

[38] S. Shekhar, C.-T. Lu and P. Zhang, "Detecting Graph-based Spatial Outliers: algorithms and applications (A summary of results)". In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 371 – 376, 2001.

[39] A. Tung, J. Hou and J. Han, "Spatial Clustering in the Presence of Obstacles". In *Proceedings of the 17th International Conference on Data Engineering*, pages 359 – 367, April 2001.

[40] W. Wang, J. Yang and R. Muntz, "Sting: a Statistical Information Grid Approach to Spatial Data Mining". In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, pages 186 – 195, 1997.

[41] G. Williams, R. Baxter, H. He, S. Hawkins and L. Gu, "A Comparative Study for RNN for Outlier Detection in Data Mining". In *Proceedings of the 2nd IEEE International Conference on Data Mining*, page 709, Maebashi City, Japan, December 2002.

[42] X. Xu, M. Ester, H.-P. Kriegel, J. Sander, "A Distribution-based Clustering Algorithm for Mining in Large Spatial Databases". In *Proceedings of the 14th International Conference on Data Engineering*, pages 324 – 331, February 1998.

[43] K. Yamanishi and J. Takeuchi, "Discovering Outlier Filtering Rules from Unlabeled Data: combining a supervised learner with an unsupervised learner". In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 389 – 394, 2001.

[44] K. Yamanishi, J. Takeuchi, G. Williams and P. Milne, "On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms". In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, Volume 8, Issue 3, pages 275 – 300, May 2004.

[45] Y.-F. Zhang, J.-L. Mao and Z.-Y. Xiong, "An Efficient Clustering algorithm". In *Proceeding of the Second International Conference on Machine Learning and Cybernetics*, Volume 1, pages 261 – 265, November 2003.

[46] J. Zhang and J. Modestino, "A Model-Fitting Approach to Cluster Validation with Application to Stochastic Model-Based Image Segmentation". In *Proceedings of the IEEE International Conference on Pattern Analysis and Machine Intelligence*, Volume 12, Issue 10, pages 1009 – 1017, October 1990.

[47] B. Ghosh-Dastidar and J.L. Schafer, "Outlier Detection and Editing Procedures for Continuous Multivariate Data". *ORP Working Papers*, September 2003. (http://www.opr.princeton.edu/papers/), visited 20.09.2004.

[48] J. Han and M. Kamber, “Data Mining: Concepts and Techniques”. *The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers*, 550 pages, August 2000.
(http://www.cs.sfu.ca/~han/DM_Book.html), visited 11.11.2004.

[49] KDD Cup 1999 Data,
(http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html), visited 18.10.2004.

Appendix

Dataset	error rates	HTER		
		proposed method	ODIN	MkNN
ftp	FA	9	57	55
	FR	36	0	63
ftp_data	FA	57	74	70
	FR	3	1	60
smtp	FA	25	99	99
	FR	25	0	50
others	FA	62	93	17
	FR	21	0	85
http	FA	75	99	33
	FR	1	0	97