Performance Evaluation of Density-Based Outlier Detection on High Dimensional Data

P. Murugavel
Research Scholar, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu, India

Dr. M. Punithavalli
Research Supervisor, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu, India

***Abstract*—Outlier detection is a task that finds objects that are considerably dissimilar, exceptional or inconsistent with respect to the remaining data. Outlier detection has wide applications which include data analysis, financial fraud detection, network intrusion detection and clinical diagnosis of diseases. In data analysis applications, outliers are often considered as error or noise and are removed once detected. Approaches to detect and remove outliers have been studied by several researchers. Density based approaches have been proved to be effective in detecting outliers successfully, but usually requires huge amount of computations. In this paper, two approaches that enhance the traditional density based method for removing outliers are analyzed. The first method uses data partitioning method and use speed up strategies to avoid large computations. The second method presents a unified clustering and outlier detection using Neighbourhood based Local Density Factor (NLDF). The aim of both the models is to improve the performance of outlier detection, clustering and to speed up the whole process. In this paper, the working of these two papers is studied and a performance evaluation based on clustering efficiency and outlier detection efficiency is presented.**

Keywords : Outlier Detection, Density-based, Neighbourhood based Local Density Factor, Clustering.

I. INTRODUCTION

Data analysis industries are envisaging continuous growth in both size and number of databases that are collected worldwide. This growth is inversely proportional to the knowledge gaining techniques available, thus necessitating development of methods that either improve or propose new algorithms for efficient mining of knowledge. Data mining is defined as the art of knowledge discovery through data analysis to reveal interesting and important pattern or trend that reflects a natural phenomenon (Atluri *et al*., 2007). During analysis, often, a special pattern called ‘Anomaly’ is detected, which is a pattern of data that does not conform to the expected behaviour. Anomaly is also referred as outliers, exceptions and peculiarities. Detection of outliers is considered as an important pre-processing step by all tasks of data mining (Xiaong *et al*., 2006). It is an important tool for detecting fraud, network intrusion, financial fraud detection, clinical diagnosis of diseases, data analysis and other rare events that are significant but hard to find. Data anomality can arise due to various reasons like data entry mistakes. Non-detection of outliers often leads to inaccurate knowledge and aids in identifying, preventing, and repairing the effects of malicious or faulty behavior. Many data mining and machine learning algorithms do not perform well in the present of outliers. Moreover, according to Otey *et al*. (2005), accurate, efficient removal of outliers may greatly enhance the performance of statistical and data mining algorithms and techniques.

The problem of outlier detection has several solutions and can be grouped into 5 main categories. They are statistical-based approaches, depth-based approaches, clustering approaches, distance-based approaches and density-based approaches. The working of these techniques are reviewed and compared by Xi (2008). Statistical-based approaches for outlier detection are mainly used to solve database that follow a univariate (parametric) distribution and are generally ill-suited for even moderately high-dimensional data. Further, statistical approach also produce high false positive rate (Mahoney and Chan., 2002; Otey *et al*., 2003), computationally expensive. To solve the problem of statistical methods, the other methods were proposed. Depth-based approaches are based on computational geometry and computes different layers of k-d convex hulls and flags objects in the outer layer as outliers (Johnson *et al*., 1998). However, it is a well-known fact that the algorithms employed suffer from the dimensionality curse and cannot cope with large k value.

When the number of normal attributes is more than the abnormal behavioural attributes, a clustering-based approach to outlier detection provides more positive results. In these situations, the key assumption made here is that large and dense clusters have normal data and the data which do not belong to any cluster or small clusters (low dense clusters) are considered as outliers. The clustering algorithms use the distance measure between two objects and clustering is performed by grouping objects which have minimum distance from the centre of the cluster. The advantage of using cluster-based algorithm is that they are easily adaptable to incremental mode suitable for anomaly detection from temporal data. On the other hand, they are computationally expensive and large/dense clusters frequently have both inliers and outliers (Guha *et al*., 2000).

In distance based approaches (Knorr *et al*.12, 13, 11), an object 'o' in a data set 'T' is a distance-based outlier, if at least a fraction 'p' of the objects in 'T' are further than distance 'D' from 'o'. These kind of outliers are based on a single, global criterion determined by the parameters 'p' and 'D'. Problems may occur if the parameters of the data are very different from each other in different regions of the data set (Sequeira and Zaki, 2002).

Recently, density-based approaches to outlier detection have been proposed (Breunig *et al*., 2002). In this approach, a local outlier factor (LOF) is computed for each point. The LOF of a point is based on the ratios of the local density of the area around the point and the local densities of its neighbors. The size of a neighborhood of a point is determined by the area containing a user-supplied minimum number of points (MinP ts). A similar technique called LOCI (Local Correlation Integral) is presented in (Papadimitriou *et al*., 2003). LOCI addresses the difficulty of choosing values forMinP ts in the LOF-based technique by using statistical values derived from the data itself. Both the LOF- and LOCI-based approaches do not scale well with a large number of attributes and data points, and so are not considered in this evaluation.

All these approaches have it own advantages and disadvantages. Density based approaches have been proved to be effective in detecting outliers successfully, but usually requires huge amount of computations. In this paper, two methods, which has enhanced the performance of density based outlier detection methods on high dimension datasets are discussed and analyzed. Yang and Huang (2008) proposed a modified method that used data partitioning method and presented speedup strategies to avoid large computations during outlier detection. Similarly, Tao and Pi (2009) proposed an algorithm called DBCOD which unified density based clustering and density based outlier detection as an unifying process using a concept called Neighbour-hood-based Local Density Factor (NLDF). The aim of both the models is to improve the performance of outlier detection, clustering and to speed up the whole process. The remaining chapters are organized as follows: Section 2 describes the concept behind Yang and Huang (2008) model, which will be referred as YHOD Model in this paper. Section 3 describes the methodology followed by DBCOD. Experiments were conducted to evaluate the performance of the two systems and the results obtained are given in Section 5. The work is concluded with future research directions in Section 6.

II. YHOD MODEL

The YHOD model consists of two phases, namely, partitioning the dataset and detecting outliers by DBOM with speedup strategy in each subset. This section explains each steps.

A. Partitioning Algorithm

Traditional DBOM algorithm can find outliers on the sample space with arbitrary shapes. Assume that C is the core object in dataset $D \subseteq R^d$ and ε is its neighborhood radius. Given an object $o \in D$ and a number m, for every $C \in D$, if o is not within the ε–neighborhood of C and $|o_{\varepsilon\text{-set}}| \leq m$, o is called the density based outlier with respect to ε and m. Given any object P in dataset D and integer m, DBOM first computes all neighborhood objects in the ε–neighborhood of P and judge whether it is a core object to find all outliers. If the ε–neighborhood of an object within the ε–neighborhood of P includes objects whose number is more than m, it means that there is a core object within the ε–neighborhood of P. Thus P is not an outlier; otherwise, P is considered as an outlier. The iteration will not terminate until all the objects in D are processed. The computation complexity of DBOM is O(n2), where n is the number of objects in D.

As mentioned, all objects have to be loaded into memory for calculating the distance between objects to judge whether an object is within the ε–neighborhood of the given objects. In large dataset, however, the memory requirements for data storing and computing will be significant and difficult to predict and the performance of DBOM quickly becomes untenable. Data partitioning is an effective technology to solve the problem. Furthermore, due to the use of global threshold neighborhood radius and density in DBOM, it is difficult to achieve a better mining result on the dataset with inconsistent distribution. So the dataset can be partitioned into several subsets and assign different thresholds for each subset. In addition, data partitioning is conducive to the implementation of parallel computation, which will greatly increase the speed of mining. During data partitioning, the entire dataset is divided into a number of local subsets according to the sample distribution characteristics to ensure that the data distribution in local subset is as identical as possible. Then, the appropriate neighborhood radius is calculated and implement DBOM algorithm in each subset. By analyzing the obtained local outliers and their adjacent objects, one can determine whether they are actual outliers. Since the local neighborhood radius is used for outlier detection, the poor effect brought by using global neighborhood radius is eliminated.

B. Speed up process

Assume that D is a dataset whilst C is a core object with respect to ε and m in D, $C_{\varepsilon\text{-set}}$ denotes the ε–neighborhood set of C. Then for every $P \in C_{\varepsilon\text{-set}}$, the condition $d(C, P) \leq \varepsilon$ is satisfied, where d(C, P) is the Euclidean distance between P and C. It means that P is within the ε-neighborhood of C and it is not an outlier. Thus, for every data object, the need for checking whether there are core objects in the ε–neighborhood is avoided. If P is a core object, then all data in its ε–neighborhood will be marked as non-outliers. Otherwise, an analysis is performed to see where some core objects exist in ε–neighborhood of P. If exist, then P is in the ε–neighborhood of core object and therefore it is not an outlier. This remarkably reduces the computational cost.

C. YHOD Process

The YHOD process consists of the steps given in Figure 1: Traditional DBOM algorithm need calculate the distance between data objects in order to determine that whether an object is within the ε–neighborhood of another. In this model, the module of each object in dataset is calculated as a pre-processing step. If $|M(P)-M(Q)|> \varepsilon$, then the distance between P and Q (Steps 2 and 3) need not be calculated. In addition, the algorithm effectively minimizes the number seeds to identify outliers. Given any object C within a ε–neighborhood, its flag will be checked firstly. If it is a core object (flag=1), all objects within the ε–neighborhood of C will not be outliers. Thus, C need not be selected as a seed to calculate the number of its ε–neighborhood (step 5).

Step 1: Read in module table from memory, initialize ε and set variable flag to 0.

Step 2 : Assume that current data object being processed is P. For $Q \neq P$, calculate $|M(P)- M(Q)|$.

Step 3 : If $|M(P)- M(Q)|> \varepsilon$, we do nothing but select next Q. Otherwise, goto step 4.

Step 4 : Calculate d(P,Q). If Q is within the ε–neighborhood of P, the number of Pε-set plus 1.

Step 5 : For every $C \in$ Pε-set, if the flag of C is 1, process is broken; Otherwise, judge whether C is a core object.

Step 6 : If C is a core object, set its flag to 1; otherwise, set flag to 0.

Step 7 : If flag=0, it means P is a outlier and it will be output; Otherwise, go to step 2.

Figure 1 : YHOD Algorithm

The algorithm for applying algorithm concentrating at outlier detection in large detection is presented in Figure 2. First of all, sample on the entire dataset and obtain the statistical data distribution characteristics projected on each dimensionality respectively. Then determine to partition on which dimensionality and the number of subset based on the statistical results so as to make the data density distribution as much uniform as possible. The statistical analysis of data distribution can utilize the histogram method.

Step 1 : Sample on the dataset.

Step 2 : Statistically analyze on all dimensionalities and choose partition points on a desirable dimensionality.

Step 3 : Divide the data into several subsets using the partition points obtained in step 2.

Step 4 : For each partition, call the modified version of DBOM mentioned above.

Figure 2 : YHOD for high dimensional data

The computation complexity of this algorithm is $O(n_1 \log n_1 +...+ n_k \log n_k)$, where k is the number of data partitioning and $n_i (1 \leq i \leq k)$ is the number of objects in partition i.

III. DBCOD ALGORITHM

This section explains the methodology used by DBCOD algorithm. First the basic notations used are given, followed by a short description of NLDF. The algorithm DBCOD that uses NLDF for fast outlier detection is then presented.

A. *Basic Notations*

Let D be a database of size n, let p, q, o, and o' be some objects in D, and let k be a positive integer. The Euclidean distance between objects p and q is denoted as dist(p, q). The local density of p is defined using Equation (1). The neighborhood-based density factor of p is defined as $NDF_k(p)=|RkNB(p)| / |kNB(p)|$. The neighborhood-based local density factor of p is defined as $NLDF_k(p)=LD_k(p)\times NDF_k(p)$.

$$LD_k(p) = \frac{\sum_{q\in kNB} \frac{1}{dist(p,q)}}{|kNB(p)|} \qquad (1)$$

Given two objects p and q in D, p is directly neighborhood-based density reachable (DNBD reachable) from q with respect to. k, if (1) $NLDF_k(q) \geq thNldf$ (thNldf is the threshold of NLDF), and (2) $p \in kNB(q)$. Similarly, given two objects p and q in D, p is neighborhood-based density reachable (NBD reachable) from q with respect to k, if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = p$, $p_n = q$ such that p_i is DNBD reachable from p_{i+1} with respect to k. Again, given two objects p and q in D, p and q are neighborhood-based density connected (NBD connected) with respect to k, if (1) p is NBD reachable from q with respect to k or (2) q is NBD reachable from p with respect to k or (3) there is a third object o such that p and q are both NBD reachable from o with respect to k. A non-empty subset C of D is a density-based cluster with respect to k if C satisfies the following two conditions: (1) for p and q in C, p and q are NBD connected with respect to k; (2) if $p\in C$ and q is NBD connected from p with respect to k, then $q\in C$. An object o is a density-based outlier with respect to k if it is not included in any density-based cluster. Moreover, its NLDF value denotes the degree of being an outlier.

B. *NLDF*

The neighborhood-based local density factor (NLDF) is adapted from NDF in NBC [4]. The difference between NDF and NLDF is the first product item LD in NLDF. Since many different objects own the same NDF value, NDF values can not differentiate the objects well. To avoid this shortcoming, the product item LD is added.

C. *DBCOD Algorithm*

The DBCOD algorithm is given in Figure 3.

IV. EXPERIMENTAL RESULTS

In this section we evaluate the performance of the two algorithms under consideration. Both the algorithms were developed in Matlab 7.3 and were tested using a Pentium IV machine with 512MB RAM. The synthetic datasets used by Karypis *et al*. (1999) were used to test the algorithms. The experiments were conducted with four different data sets containing points in two dimensions whose geometric shape are shown in Figure 4.

The first data set, DS1, has six clusters of different size, shape, and orientation, as well as random noise points and special artifacts such as streaks running across clusters. The second data set, DS2, has nine clusters of different shape, size, and orientation, some of which are inside the space enclosed by other clusters. Moreover, DS2 also contains random noise and special artifacts, such as a collection of points forming vertical streaks. The third data set, DS3, has eight clusters of different shape, size, density, and orientation, as well as random noise. A particularly challenging feature of this data set is that clusters are very close to each other and they have different densities. Finally, the fourth data set, DS4, has six clusters of different shape connected by density chain of points. The size of these data sets ranges from 8000 to 10000 points, and their exact size is indicated in Figure 4. Figures 5 and 6 show the datasets from Figure 4 after determining the outliers (the gray points) for both algorithms. Figure 7 and 8 shows the clusters discovered in each data set by both algorithms. Table I presents the execution time of both the algorithms.

```
//Step1: Build a memory index
1.  indexFile = BuildMemoryIndex(strIdxFileName);
2.  //Step2: Compute NLDF and its threshold thNldf
3.  QuerykNBAndRkNB(indexFile, SetOfObjects, &kNB, &RkNB,&ld,k);
4.  CalculateNldf(SetOfObjects, kNB, RkNB,ld, &thNldf);
5.  //Step3: Discovering density-based clusters and finding density-based
6.  outliers
7.  FOR each object obj in SetOfObjects DO
8.      IF obj.getClusterId() != UNCLASSIFIED || obj.NLDF<thNldf THEN Continue;
9.      obj.setClusterId(clusterCount),
10.     C[clusterCount].add(obj); //Labeling a new cluster
11.     //Creating and initializing a queue and path array
12.     09: FOR each object knbObj in kNB(obj) DO
13.         knbObj.setClusterId(clusterCount);
14.         C[clusterCount].add(knbObj);
15.         IF knbObj.NLDF >= thNldf THEN //Adding to queue path[rear++] = knbObj.id;
16.     END FOR;
17.     //Expanding the current cluster
18.     WHILE (front != rear) DO
19.         frontId = path[front++];
20.         FOR each object knbObj in kNB(SetOfObjects[frontId]) DO
21.             IF knbObj.getClusterId() != UNCLASSIFIED THEN continue;
22.             knbObj.setClusterId(clusterCount);
23.             C[clusterCount].add(knbObj);
24.             IF knbObj.NLDF >= thNldf THEN path[rear++] = knbObj.id;
25.         END FOR;
26.     END WHILE;
27.     clusterCount++;
28. END FOR;
29. //Step4: Sorting density-based outliers
30. FOR each object obj in SetOfObjects DO
31.     IF obj.getClusterId() == UNCLASSIFIED THEN O.add(obj);
32. END FOR;
33. Ascending sort all outliers in O according to NLDF value
```

Figure 3 : DBCOD Algorithm

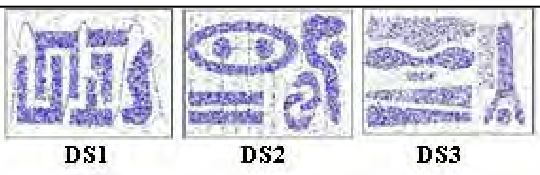

Figure 4 : Datasets used

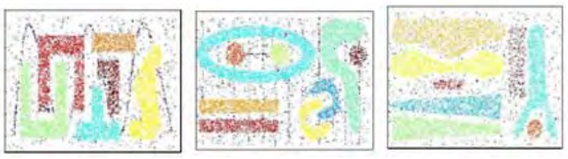

Figure 5 : Datasets after outlier detection - DBCOD

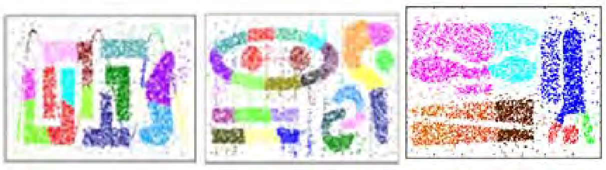

Figure 6 : Datasets after outlier detection - YHOD

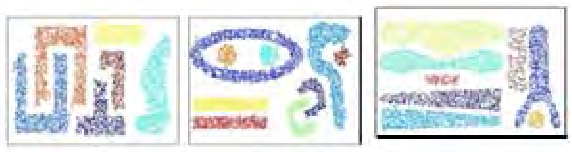

Figure 7 : Final Results – DBCOD

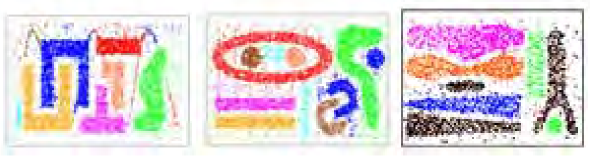

Figure 8 : Final Results - YHOB

TABLE I. EXECUTION TIME (Seconds)

Dataset	DBCOD	YHOB
DS1	0.74	0.92
DS2	0.81	1.01
DS3	1.67	2.01
DS4	0.89	1.12

From the results, it could be seen that DBCOD model is more efficient in terms of cluster formation and speed of detecting outliers.

V. CONCLUSION

Outlier detection is a task that finds objects that are dissimilar or inconsistent with respect to the remaining data. It has many uses in applications like fraud detection, network intrusion detection and clinical diagnosis of diseases. Using clustering algorithms for outlier detection is a technique that is frequently used. The clustering algorithms consider outlier detection only to the point they do not interfere with the clustering process. Out of the many methods available, density based approaches have been more efficient in detecting outliers. The main drawback of such methods is the heavy computation involved, when large datasets are used. In this paper, two algorithms, DBCOD and YHOD, were selected. Both algorithms enhance the traditional distance-based approach by reducing the computation capacity. Performance of these two algorithms in terms of outlier detection and clustering efficiency with high dimensional data were studied. Experimental results proved that DBCOD algorithm is more efficient. Recently, hybrid systems, which combine the positive aspects of one or more algorithms is frequently developed in many areas including outlier detection. In future, research is planned to combine these two algorithms with distance based methods to detect outlier during classification. The effect of the same on computation complexity and classification efficiency can also be studied.

REFERENCES

[1] Atluri, V., Adam, N.R. and Janeja, V. (2007) Anomaly detection in heterogeneous datasets, Doctoral Dissertation, ACM Digital Library, Rutgers University, New Jersey.

[2] Breunig, M.M., Kriegel, J., Ng, R.T. and Sander, J. (2002) LOF: Identifying density-based local outliers, ACM SIGMOD Intl. Conf. Management of Data.

[3] Guha, S., Rastogi, R. and Shim, K. (2000) ROCK: A robust clustering algorithm for categorical attributes, Information Systems, Vol. 25, No.5, Pp.345–366.

[4] Johnson, T., Kwok, I. and Ng, R. (1998) Fast computation of 2-dimensional depth contours, Proc. KDD, Pp. 224–228

[5] Karypis, G., Han, E.H. and Kumar, V. (1999) CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling, Computer,Vol. 32, pp. 68-75.

[6] Mahoney, M.V. and Chan, P.K. (2002) Learning nonstationary models of normal network traffic for detecting novel attacks, ACM SIGKDD.

[7] Otey, M.E., Parthasarathy, S. and Ghoting, A. (2005) An Empirical Comparison of Outlier Detection Algorithms, DMMAD Workshop Notes, Pp. 45-52

[8] Otey, M.E., Parthasarathy, S., Ghoting, A., Li, G. , Narravula, S. and Panda, D. (2003) Towards nic-based intrusion detection, Proceedings of 9th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[9] Papadimitriou, S., Kitawaga, H., Gibbons, P.B. and Faloutsos, C. (2003) LOCI: Fast outlier detection using the local correlation integral, ICDE.

[10] Sequeira, K. and Zaki, M. (2002) Admit: Anomaly-based data mining for intrusions, ACM SIGKDD 02, Pp. 386–395, 2002.

[11] Tao, Y. and Pi, D. (2009) Unifying Density-Based Clustering and Outlier Detection, Second International Workshop on Knowledge Discovery and Data Mining, IEEE Computer Society, Pp.644-647.

[12] Xi, J. (2008) Outlier Detection Algorithms in Data Mining, Second International Symposium on Intelligent Information Technology Application, Vol. 1, Pp.94-97.

[13] Xiong, H., Pandey, G., Steinbach, M. and Kumar, V. (2006) Enhancing Data Analysis with Noise Removal, IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 3, Pp. 304-319.

[14] Yang, P. and Huang, B. (2008) A Modified Density Based Outlier Mining Algorithm for Large Dataset, International Seminar on Future Information Technology and Management Engineering, IEEE Computer Society, Pp. 37-40.