Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream

Manzoor Elahi[1,2,3] Kun Li [1,2] Wasif Nisar[1,3] Xinjie Lv [1,2]
Hongan Wang [1,2]
[1]Intelligence Engineering Lab, Institute of Software Chinese Academy of Sciences, Beijing, 100190
[2]Graduate University of Chinese Academy of Sciences, Beijing, China, 100049
[3]COMSATS institute of information Technology, Islamabad, Pakistan
tamimy@gmail.com

Abstract

Anomaly detection is currently an important and active research problem in many fields and involved in numerous applications. Most of the existing methods are based on distance measure. But in case of Data Stream these methods are not very efficient as computational point of view. Most of the exiting work on outlier detection in data stream declare a point as an outlier/inlier as soon as it arrive due to limited memory resources as compared to the huge data stream, to declare an outlier as it arrive often can lead us to a wrong decision, because of dynamic nature of the incoming data. In this paper we introduced a clustering based approach, which divide the stream in chunks and cluster each chunk using k-mean in fixed number of clusters. Instead of keeping only the summary information, which often used in case of clustering data stream, we keep the candidate outliers and mean value of every cluster for the next fixed number of steam chunks, to make sure that the detected candidate outliers are the real outliers. By employing the mean value of the clusters of previous chunk with mean values of the current chunk of stream, we decide better outlierness for data stream objects. Several experiments on different dataset confirm that our technique can find better outliers with low computational cost than the other exiting distance based approaches of outlier detection in data stream.

1. Introduction

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. A data stream is an ordered sequence of objects $X_1, .., X_n$. The main difference between a traditional database and a data stream management system (DSMS) is that instead of relations, we have unbounded data streams. Applications, such as fraud detection, network flow monitoring, telecommunications, data management, etc., where the data arrival is continuous and it is either unnecessary or impractical to store all incoming objects.

Anomaly detection deals with detecting data elements from a data set which is different from all the other data elements in a set. Anomalies can arise due to different reasons such as mechanical faults, other changes in the system, fraudulent behavior, instrument error, human error or natural deviation. Usually anomalous observations are more interesting and need excess examination. It is not easy to define exactly what is an anomaly or an outlier.

Traditional methods for outlier detection can produce good results on stored static dataset. Traditional data mining methods cannot be applied to streaming data efficiently as these methods are suitable for the environment where the entire dataset is already available and algorithm can operate in more than one pass. A general framework for mining data streams need small constant time per record along with the minimum memory requirement, using at most one scan of data. As the nature of data stream is unbounded the problem of mining outlier in data streams is often performed based on certain times intervals, usually called windows.

Recently, some data mining methods [27, 28] for a data stream have been actively introduced. In the traditional database environment there is a lot of research towards outlier/anomaly detection. Among those Distance based approach first proposed by Knorr and Ng [12] and later this approach is further extended in different scenarios. All these techniques are highly dependent on the parameters provided by the users which is difficult in increasing dimensionality, and the wrong choice can effect the results. Later LOF [3]

978-0-7695-3305-6/08 $25.00 © 2008 IEEE
DOI 10.1109/FSKD.2008.374

and its extension [25] although useful one, the computation of LOF values for every data objects require a large number of nearest neighbor searches.

The existing methods are either nearest neighbor based, which calculate the pairwise distances in a specified neighborhood, which doesn't suit in the environment of data stream, or model based, which are specific for intrusion detection in network based data flow. Where a profile or model is developed first and then incoming data is compared with the model to detect abnormal data. Few researchers applied same techniques and methods for streaming data, but most of the methods are computationally expensive if applied to unbounded data streams.

Moreover due to stream evolution, object properties can change over time hence, evaluating an object for outlierness when it arrives, although meaningful, often can lead us to a wrong decision, because of dynamic nature of the data stream. With the help of following example we will point out the problem in most of existing methods for data stream. Two diagrams shown in Figure 1 which shows the evolution

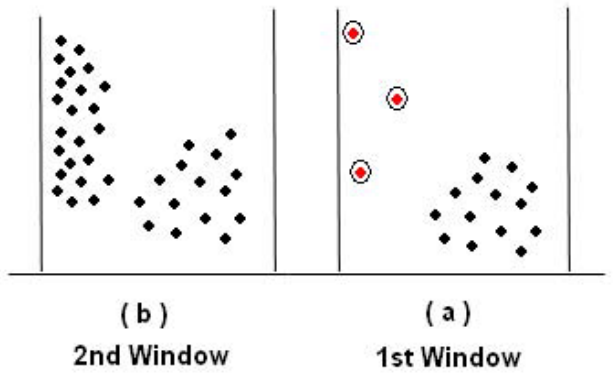


Figure 1. Outlier detection over DataStream

of data stream. We will refer to these two diagrams as 1st window and 2nd window. During the processing of 1st window three points are outliers. Most of the existing methods detect these points as outlier just by considering the current window at the time. But as in data stream the data distribution may change as the stream evolves. Points which are declared as outliers at the time of 1st window, may belongs to a dense region with the evolution of 2nd window. So here we can see that three outliers detected in 1st window actually belong to a dense region with the evolution of 2nd window. One of the recent attempt to solve this problem is presented in [27]. Author proposed two algorithms, to detect outlier they use k-nearest neighbor approach, where user has to provide two parameters R and K, an object is distance based outlier if less than k objects in S lie within distance R from it.

In brief, to avoid pairwise distance calculations, to detect better outlier even if the evolution of data stream change, to let user free to provide sensitive parameters, and to mine data stream even in limited memory resources here we propose a clustering based method because, the clustering methods have good space and time complexity.

To address these four most important issues we propose a k-mean clustering based anomaly detection approach, which divide a stream in chunks of data and cluster these chunks in fixed number of ***k*** clusters using k-mean. Instead of keeping only the summary information, which often used in case of clustering data stream, we keep the candidate outliers and mean value of every cluster for the next fixed number of steam chunks, to make sure that the detected candidate outliers are the real outliers. By employing the mean value of the clusters of previous chunk with mean values of the current chunk of stream, we decide better outlierness for data stream objects. if the difference between an object and its closest cluster center becomes large then this object will be declared as a candidate outlier and will be clustered again with new chunk of data. Our proposed method use k-mean clustering in an incremental fashion which can solve all these issues mentioned above.

With the help of several experiments on different dataset the effectiveness of the proposed method is shown.

The rest of this paper is organized as follows. Section 2 related work is presented, Section 3 formally represent the outlier detection problem which deals with our work. Section 4 describe proposed algorithm in detail. Section 5 illustrate Experimental results. Finally Section 6 presents conclusion.

2. Related Work

There is a lot of literature on the anomaly detection problem which describes a variety of approaches like, Distance-Based outlier Detection is proposed by Knorr and Ng [12]. Given parameters k and R, an object is a distance-based outlier if less than k objects in the input data set lie within distance R from it. Further extended by Ramaswamy et al in [2] having idea, in order to rank the outliers, introduced the following definition: given k and n, an object o is an outlier if no more than n-1 other objects in the dataset have higher value for Dk than o, where Dk(o) denotes the distance of the k^{th} nearest neighbor of o. This concept is future extended in [7, 18, 19], where each data point is ranked by the sum of distances from its k nearest neighbors. Distance based outliers are not suitable if the clusters have different densities so to overcome the shortcoming of distance based outliers Breunig et al proposed a concept of LOF [3] which are the objects outlying relative to their local neighborhoods, with respect to densities of the neighbor-

hood. This concept is useful but to compute the LOF value large number of k-nearest neighbor searches make it computationally expensive. On the bases of observing density distribution from the data, Aggarwal and YU [8] proposed a technique for outlier detection. The basic idea in their definition is, a point is an outlier, if in some lower dimensional projection it is present in a local region of abnormally low density. This method is also an efficient method for high dimensional data set. Some Clustering-Based outlier detection techniques are proposed [9, 20]. This technique work in two basic steps, fixed width clustering with w(radius) and after the first phase the next step is sorting of clusters produced in the first step. Points in the smaller clusters are declared as outliers. Clustering-Based techniques are further extended by proposing the concept of cluster-based local outlier, in which a measure for identifying the outlierness of each data object is defined. Deviation-based techniques identify outliers by inspecting the characteristics of objects and consider an object that deviates these features, declared as an outlier [20]. Distance-based methods previously discussed are designed to work in the batch framework, under the assumption that the whole data set is stored in secondary memory and multiple passes over the data can be accomplished. Hence, they are not suitable for data streams. While the majority of the approaches to detect anomalies in data mining consider the batch framework, some researchers have attempted to address the problem of online outlier detection. The replicator neutral network (RNN) based technique is introduced to detect outliers by Harkins, et al. [22]. Although there is lot of research on outlier detection, but there is little research in the direction of outlier detection in dynamic data streams. This area still needs lot of attention because the existing methods are not appropriate in the stream environment.

There is an abundant literature addressing Data Streams, but their main focus is to solve the problem of clustering [13, 10], query processing [14], frequent pattern mining [15, 16] but they also didn't address the problem of anomaly detection in Data Streams.

Clustering techniques are categorized into several different methods: partitioning, hierarchical [5], a pair of initial clusters are successively merged until the predefined number of clusters is left. The density-based DBSCAN[6] and CLIQUE [21] regards a cluster as a region in a data space with the proper density of data elements. Clustering on a data stream is categorized by kmeans/k-median and grid-based methods. In order to identify the clusters of objects occurring in a data stream, a k-median algorithm is proposed [4]. It regards a data stream as a sequence of stream chunks. In the grid-based method [10], the data space of the feature is partitioned into a set of mutually exclusive equal-size initial cells. As a new user activity is performed continuously, each initial cell monitors the distribution statistics of its corresponding feature values within its range. Eventually, a dense region of each initial cell is recursively partitioned until it becomes the smallest cell called a unit cell.

3. Problem Definition

3.1 problem statement

First of all let us define some formal definitions of k-mean and DataStream.

Definition 1: *Let an integer k and a set S of points in Euclidean space be given. For any set of k points* $P = \{c_1, c_2, ..., c_k\}$ *in the space, S can be partitioned into k corresponding clusters* $C_1, C_2, ..., C_k$ *by assigning each point in S to the closest center* c_i*. The problem of k-means is to find the set P such that the sum of squared distance (SSQ) measure, defined as*

$$\sum_{i=1}^{k} \sum_{p \in Ci} |p - c_i|^2$$

is minimized. [29]

A DataStream $DS = \{x_1, x_2, ..., x_n\}$ possibly infinite series of objects. x_i is represented by n-dimensional vector i.e., $x_i = (x_i^1, x_i^2, ..., x_i^n)$ therefore data projected to he k^{th} dimension from the data stream DS is represented as

$$DS^k = x_1^k, x_2^k, ..., x_n^k$$

Let data stream objects are elements of a metric space on which we can define a distance function.

We consider a data stream as a chunks of data

$$DS = X_1, X_2, ..., X_m$$

where every chunk contains specified number of n points.

Problem : *Given a data stream* **DS***, a chunk size n, fixed number of clusters K, detection of outlier is to find a point which is far from its cluster center Until L number of chunks* where,

Let C^j denote the group of clusters of j^{th} chunk of stream.

$$C^j = C_1^j, C_2^j, .., C_k^j \; 1 \leq j \leq m$$

Every C_p^j contains two mean values denoted as $C_p^j\mu$A, $C_p^j\mu$U. where $C_p^j\mu$A represent the actual mean of cluster C_p^j, while $C_p^j\mu$U represent Updated mean of cluster C_p^j.

μ represents the central value of cluster, can be calculated as

$$\mu = \frac{1}{N} \sum_{l=1}^{r} x_l$$

where N represent the total number of elements in C_p^j where $C_p^j \in C^j$.

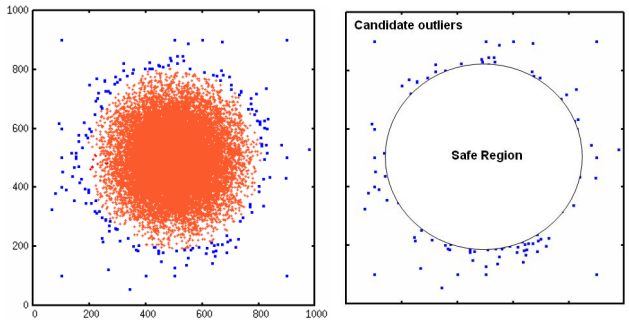


Figure 2. Basic Idea of proposed method

4. Proposed Method in Detail

4.1 Motivation

In order to deal with the problem of processing data stream even in limited memory resources, the proposed method can give solution by keeping the mean values and temporary candidate outliers and discarding the safe region. By this way we can free some memory for the next chunk of data to be accommodated. The basic assumption for the outlier detection is that the data in sparse region can be a better candidate for outlier.

Declaring an object as an outlier as it arrive, can often lead us to a wrong decision. In order to deal with this problem in the proposed method objects will not be declared as outlier after processing of current window or chunk of data, instead, the objects found different than the rest of the data will be declared as temporary candidate outliers for the next fixed number of stream chunks.

Most of the existing work use k-nearest neighbor approaches for outlier detection which need parameters R and K, to be provided by the user, is often difficult for the user to select these parameters. Moreover these methods perform pairwise distance calculations which can become computationally expensive for data stream. While in proposed method using clustering, we do not need pairwise distance calculations.

4.2 Overview of proposed Algorithm

For our proposed method first we need a clustering algorithm. In this paper, the clustering algorithm used is the K-mean algorithm, which can produce good clustering results and at the same time deserves good scalability. For the process of mining outliers in our proposed method, the clustering algorithm used for grouping chunks of incoming stream into disjoint sets of records can be chosen freely.

First Partition X in K clusters using K-mean, after clustering figure out candidate outliers from every cluster, if the difference between the points and its respective cluster center becomes large compared to the rest of the objects. In the next step, keep two mean values for every cluster denoted as, Actual Mean, and Updated Mean and discard the most safe region on the basis of deviation criteria used to detect candidate outliers. By discarding the safe region we will be able to accommodate next chunk of data. Compare candidate outliers in next L number of chunks. Declare candidate outliers as inlier or outlier if their outlier score becomes equal to user specified limit L. By using proposed method, we can even process a data stream in very limited memory resources.

Algorithm 1 Cluster based OutlieR Miner(CORM)

Require: K: Number of clusters
Require: L: Till how many chunks to test Outlierness
Require: $X_j = \{x_1, x_2, x_3, ..., x_n\}$
Require: n: Chunk size
if $Partitons = NULL$ **then**
  cluster(X_j, K) partition the data into fixed number of clusters with new Initial centers.
else
  Distribute data in K clusters using cluster centers of previous chunk.
  $C_p^j\mu$A= $\frac{1}{N}\sum_{l=1}^{r} x_l$, where, $1 \leq p \leq K$
  $C_p^j\mu$U= $(C_p^j\mu$A+$(C_p^{j-1}\mu$U$))/2$
  $cluster(X_j, K)$ cluster data using updated mean.
end if
Mark points as ***candidate outliers*** on the basis of deviation criteria used to detect candidate outliers.

$$dist(C_p^j, x_i) = \left|C_p^j\mu U - x_i\right|$$

Keep $C_p^j\mu$A, $C_p^j\mu$U, candidate outliers, of each cluster.
Discard the safe Region
Assign Score of survival to every ***candidate outliers*** in there respective cluster on the basis of count, where $1 \leq count \leq L$.
if $OutlierScore(Candout^j) = L$ **then**
  if $dist(C^j, Candout^j) > C_p^j\mu$U **then**
    Declare $Candout^j$ as real outlier
  else
    Declare $Candout^j$ as inlier and remove from the list.
  end if
end if
if Next chunk of stream EXIST **then**
  goto step 1
end if
$Exit.$

4.3 Algorithm Description

Let us assume that a DataStream is represented as $DS^k = \{x_1^k, x_2^k, ..., x_n^k\}$ and if we consider a DataStream as a chunks of data then it can be represented as $DS = X_1, X_2, ..., X_m$, where every chunk contains fixed number of m object.

Let C^j denote the group of clusters of j^{th} chunk of stream. $C^j = C_1^j, C_2^j, .., C_k^j$ where $1 \leq j \leq m$ Every C_p^j contains two mean values denoted as $C_p^j\mu$A, $C_p^j\mu$U, where $C_p^j\mu$A represent the actual mean of cluster C_p^j, while $C_p^j\mu$U, represent Updated mean of cluster C_p^j.

μ represents the central value of cluster, can be calculated as $\mu = \frac{1}{N}\sum_{l=1}^{r} x_l$

Where N is total number of elements in C_p^j where $C_p^j \in C^j$ In the proposed algorithm the first step is we input Chunks of data one after the other, $X_j = \{x_1, x_2, x_3, ..., x_n\}$ where X_j represent j^{th} chunk while x_i is the i^{th} element in the chunk X_j. Procedure for k-mean clustering is simple, partition the dataset into k subsets. In the proposed method we partition X_j in to specified number of K groups, called clusters. Which produce K center points for these clusters, each representing an individual cluster. If the partition are empty, if no clusters exist, the simple clustering will be adopted by choosing an appropriate cluster center and assigning all the objects in the current chunk to their nearest cluster. If the partitions are not NULL then the objects of the chunks are first assigned to the nearest clusters centers on the basis of cluster centers produced in previous chunk i.e $C_p^{j-1}\mu$U where $1 \leq p \leq K$. Next step is calculation of $C_p^j\mu$A, and updating of $C_p^j\mu$U. $C_p^j\mu$A new mean value for clusters in X_p and update $C_p^j\mu$U, by combining the new mean with previous mean values of the clusters produced in L number of chunks using the additive property. where L is the parameter defined by the user i.e. for how many chunks of streams a user want to confirm the outlierness of a candidate outlier. Candidate outliers found from previous L number of chunks are also clustered, but they are not contributing in the calculation of $C_p^j\mu$A for the sake of better understanding we give an example at the end of this section.

Outlier score/degree for every candidate outlier is incremented as it move to the next chunk of stream for processing. Two mean values, Actual mean $C_p^j\mu$A, and updated mean calculated using additive property i.e. if a cluster C_1^2 contain points $x_1, x_2, x_3, ..., x_n, o1, o2, o3$ then the actual mean is value obtained through $x_1, x_2, x_3, ..., x_n$.

Moreover, we must make decision about the candidate outliers at stage L, and the value of parameter L need to be selected carefully, by keeping in view the values of parameter K and n.

4.4 Example

Let $DS = X_1, X_2, ..., X_m$ during the clustering phase of X_1 we cluster X_1 in fixed number of cluster say $K = 3$, C_1^1, C_2^1, C_3^1 and find some points (o1,o2,o3) as candidate outliers from cluster C_1^1. After finding the points (o1, o2, o3), we discard rest of the data in cluster C_1^1. Now input 2nd chunk X_2, cluster the data in the same fixed number of K clusters. Update and keep the mean values.

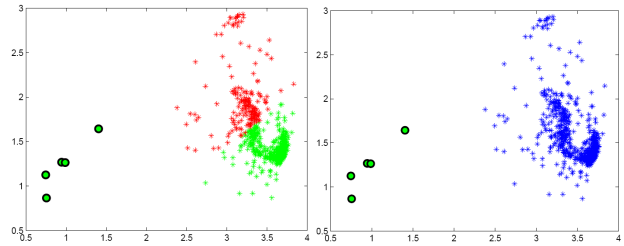
At this stage we find some candidate outliers (o1, o2, o3, o4) where (o1, o2, o3) declared as candidate outliers from C_1^1 also remain outlier during C_1^2 because they still deviate from the the normal data, and the o4 a new outlier found in C_1^2. We move to next chunk X_3. If the user specified parameter L = 3 then this is the phase where we decide a point to be an inlier or outlier, i.e at the end of the processing phase of X_3 we will make decision for the candidate points found in C_1^1, same at C_1^4 we make decision for C_1^2 and so on. At the stage of X_3 we must make decision for points found as candidate outliers at X_1 and we will remove these points from the clusters whether they are inlier or outliers because the user specified parameter $L = 3$.

5. Experimental Results

We implemented our algorithm in matlab. For the proposed method we use synthetic dataset, as well as real dataset, like abalone data UCI machine learning repository [26]. Which contain 4177 examples with 8 attributes. We input a set of examples as a chunk X. Synthetic dataset used in our experiment contain 30,000 examples. We tested our algorithm with different sizes of the chunk with different number of dimensions and using different kind of data set and found that it produce better results as compared to the distance based approaches, not only efficient regarding detection of outliers, but also efficient as a computational point of view.

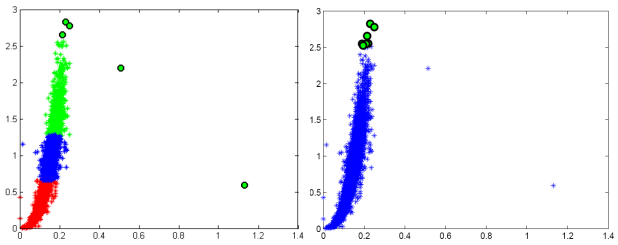
We compare our results with DK (distance based k-nearest neighbor) approach. This method based on distance of a point from its k^{th} nearest neighbor. This method rank points on the basis of distance to its k^{th} nearest neighbor and declare the top-n points in ranking as outlier. Some interesting results of our experiments on proposed method as compared to the Dk approach is shown in Figure 3, Figure 4 and Figure 5.

The test results presented in Figure 3, where we selected a mixed attribute values from abalone dataset, containing 4177 instances where the outliers were present in a sparse region. We clustered data in 2 groups and found that the test results were same, both methods found same number of outliers, in this case 4 outliers as shown in Figure 3(a) produced by proposed method while Figure 3(b) represent DK method. Although the number of outliers detected were



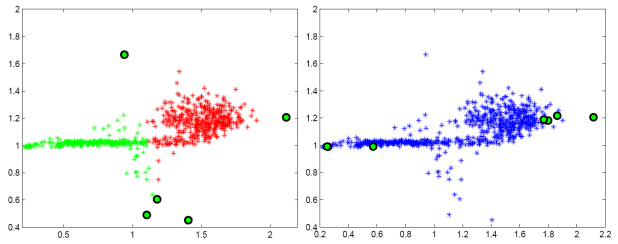
(a) Proposed (CORM) (b) Dk-Approach

Figure 3. Sparse outlier CORM vs DK-outliers comparison



(a) Proposed (CORM) (b) Dk-Approach

Figure 4. Dense-dataset CORM vs DK-outliers comparison



(a) Proposed (CORM) (b) Dk-Approach

Figure 5. Mix-dataset CORM vs DK-outliers comparison

same but there was a clear difference of computation time. As the DK approach take longer than our proposed method.

Test results of Figure 4 were based on dataset where most of the data is in dense region and the outliers were still present in sparse region. Figure 4(a) represent proposed method, we divide the data in 3 clusters, and we can see that proposed method produce much accurate results as compared to DK. DK approach left two outliers which were very far from the rest of the data and only mark those points as outliers which were not suitable to be detected as outliers Figure 4(b) shows the test results. While proposed method detect 5 most suitable outlier with a big margin of CPU time as compared to DK k-nearest neighbor approach.

For Figure 5 where the data distribution was mixed again but the outliers were present in sparse as well as near the dense region. Figure 5(a) represents the test results of proposed method, here we can see that DK approach again left the 4 most outstanding outliers, and mark some points as outliers which were not suitable to be marked as outliers. Figure 5(b) shows the test results of Dk approach. While proposed method detect 5 most suitable outlier with a big margin of CPU time as compared to DK k-nearest neighbor approach.

In case of DK-Outliers to calculate CPU time, the parameters used are $K = 3; n = 7$ where n = number of values we want to identify while K = nearest neighbors to consider in the calculus of Dk, while for Dk-Outliers Nested Loop $D = 0.45; p = 0.95$. We did comparison for many different dataset with different number of attributes, by passing more suitable parameter the difference between the CPU times of both these algorithms is shown in Table 1.

Table 1. CPU Time in seconds

N	Proposed	DK-Outliers	Dk-Outliers NL
4177	0.212416	7.906283	88.814268
8000	0.673751	34.65489	569.120207

Moreover the accuracy of detecting outlier is shown in the Figure 3(a), Figure 4(a) and Figure 5(a), which shows that, by keeping the candidate outlier in the first chunk and take it to k number of chunks, where k should not be too large, can help us to improve the accuracy of finding outliers, instead of declaring them outlier as they arrive.

6. Conclusions

In this paper, we present a cluster-based outlier detection in data stream, where we pay more attention to the points detected as outliers and give them a chance of survival in the next incoming chunk of data, rather declare them outlier by observing the current chunk. We only keep the most suitable candidate outliers. As in data stream we can't keep the entire stream in some physical memory so we discard the region which is safe and do not contain outliers, and free memory for the next generation of data to be processed effectively. The experimental results show that our approach outperform some existing methods on identifying meaningful outliers over DataStream. For future work, we will integrate Distance-Based approach more tightly with clustering algorithms and will introduce an efficient way to assign outlierness degree to the detected outliers in data streams.

References

[1] B.Mukherjee, T.L. Heberlein, and K.N.Kevitt, Network intrusion Detection, IEEE Network, 8(3):26-41, May/June 1994.

[2] Ramaswamy S., Rastogi R., Kyuseok S.: Efficient Algorithms for Mining Outliers from Large Data Sets, Proc. ACM SIDMOD Int. Conf. on Management of Data, 2000

[3] M.M. Breunig, H.P.Kriegel, R.T. Ng and J.Sander, LOF: Identifying Density-Based Local Outliers ACM SIGMOD 2000

[4] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. Clustering data streams: Theory and practice, IEEE Trans. Knowl. Data Eng 15, 3 (2003), 515–528.

[5] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, Birch: An Efficient data clustering method for very large databases, Proc. for the ACM SIGMOD Conf. on Management of Data, Montreal, Canada, June 1996.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, (KDD 96), Portland, Oregon, 1996

[7] Fabrizio Angiulli, Clara Pizzuti, Fast Outlier Detection in High Dimensional Spaces, Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, p.15-26, August 19-23, 2002

[8] Charu C. Aggarwal, Philip S. Yu, Outlier detection for high dimensional data, Proc. of the 2001 ACM SIGMOD int. conf. on Management of data, p.37-46, May 21-24, 2001, Santa Barbara, California, United States

[9] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In Data Mining for Security Applications, 2002.

[10] Nam Hun Park, Won Suk Lee, Statistical grid-based clustering over data streams, ACM SIGMOD Record, v.33 n.1, March 2004

[11] F. Grubbs. Procedures for detecting outlying observations in samples. Technometrics, 11(1):1-21, 1969.

[12] Knorr, E. M., Ng, R. T. Algorithms for Mining Distance-Based Outliers in Large Datasets, Proc. 24th VLDB, 1998

[13] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proceeings of the 29th VLDB conference, 2003.

[14] Nitin Thaper, Sudipto Guha, Piotr Indyk, Nick Koudas, Dynamic multidimensional histograms, Proc. of the 2002 ACM SIGMOD int. conf. on Management of data, June 03-06, 2002, Madison, Wisconsin

[15] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In Proceedings of the 28th Int. Conf. on VLDB 2002.

[16] Jeffrey Xu Yu, Zhihong Cheng, Hongjun Lu and Aoying Zhou, False positive or false negative: mining frequent itemsets from high speed transactional data streams VLDB 2004

[17] F. Angiulli and C. Pizzuti. Outlier mining in large high-dimensional data sets. IEEE Transaction on Knowledge and Data Engineering, 17(2):203(215, February 2005).

[18] F. Angiulli, S. Basta, and C. Pizzuti. Distance-based detection and prediction of outliers. IEEE Transaction on Knowledge and Data Engineering, 18(2):145(160, February 2006).

[19] M. F. Jiang, S. S. Tseng, C. M. Su. Two-phase clustering process for outliers detection. Pattern Recognition Letters, 2001, 22(6/7): 691-700.

[20] A. Arning, R. Agrawal, P. Raghavan. A linear method for deviation detection in large databases. In: Proc of KDD'96, 1996: 164 169.

[21] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data, Seattle, Washington, June 1998.

[22] S. Harkins, H. He, G. J. Willams, R. A. Baster. Outlier detection using replicator neural networks. In: Proc of DaWaK'02, 2002: 170-180.

[23] P. Frnti and J. Kivijrvi, Randomised Local Search Algorithm for the Clustering Problem. Pattern Analysis and Applications, Volume 3, Issue 4, pages 358 - 369, 2000.

[24] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. To appear in Proceedings of IEEE Int. Conf. on Data Engineering, March 2002.

[25] Wen Jin and Anthony K. H. Tung and Jiawei Han, Mining top-n local outliers in large databases. pages 293-298, year 2001,

[26] Asuncion, A.and Newman,D.J. "2007".UCI Machine Learning Repository Irvine,CA University of California, School of Information and Computer

[27] Angiulli, F. and Fassetti, F. Detecting distance-based outliers in streams of data. In Proc. of the Sixteenth ACM Conf. on information and Knowledge Management (Lisbon, Portugal, November 2007). CIKM '07.

[28] Pokrajac, D. Lazarevic, A.Latecki, L.J. Incremental Local Outlier Detection for Data Streams Computational Intelligence and Data Mining 07. CIDM 2007.

[29] Maria E. Orlowska, Xingzhi Sun, Xue Li: Can exclusive clustering on streaming data be achieved?. SIGKDD Explorations (SIGKDD) 8(2):102-108 (2006)