IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

Efficient Distance Based Continuous Outlier Detection for Unsupervised Dataset

Manochitra. S[1], Reshma. R[2], Safeedha P.K[3], Saranya R[4]

Assistant Professor, Department of Computer Science and Engineering, Dhaanish Ahmed Institute of Technology, Coimbatore, India[1]

Student, Department of Computer Science and Engineering, Dhaanish Ahmed Institute of Technology, Coimbatore, India[2,3,4]

Abstract: Time series data streams are common due to the increasing usage of wireless sensor networks. Such data are often accompanied with uncertainty due to the limitations of data collection equipment. Outlier detection on uncertain static data is a challenging research problem in data mining. Moreover, the continuous arrival of data makes it more challenging. In this paper propose continuous outlier detection is a special class of steam data mining. Typically, stream data mining algorithms assume that each object is inspected at most once. However, in continuous outlier detection need to be capable of reporting, at each time point, the outliers among all the objects in the current sliding window. The propose a sliding window approach of outlier detection, which makes use of the results obtained from the previous state set to efficiently detect outliers in the current state set. These methods are verified using both real data and synthetic data. The results show that they are able to reduce the required storage and running time.

Keywords: Intrusion Detection Systems, Sliding window, MCOD, Event window, sliding window.

I. INTRODUCTION

This Outlier detection is a fundamental problem in data mining. It has applications in many domains including credit card fraud detection, network intrusion detection, environment monitoring, medical sciences, etc. Mining outliers is considered an important task in many applications like fraud detection, plagiarism, computer network management, event detection (e.g., in sensor networks), to name a few. In simple terms, an object is considered an outlier, if it deviates from the "typical case" significantly. "An outlier is an observation in a data set which appears to be inconsistent with the remainder of that set of data". The process of outlier detection may be seen as the complement of clustering, in the sense that clustering tries to form groups of objects whereas outlier detection tries to spot objects that do not participate in a group.
Outlier detection algorithms can be applied to data of arbitrary dimensionality and also in general metric spaces. The only input needed (apart from its specific parameters) is a distance function to compute pair-wise distances. This means that it is not necessary to work with a multi-dimensional data set. Other data sets may be used as well (e.g., time series, graphs, DNA sequences) as long as a meaningful distance measure has been defined. Although the metric properties are well appreciated, the distance function used need not satisfy triangular inequality. However, this property is important for indexing purposes, and therefore we will make the silent assumption that the distance function used is a metric function. In many applications dealing with uncertain data streams, the generated data volume is huge, making it practically impossible to keep all acquired data in memory. Moreover, in most situations, only recent data is of interest and therefore, a sliding window is used in this proposed system. By using a sliding window, the most recent range of measurements is kept in a buffer. Each time we add a new element to the sliding window, and the oldest element in the window should be deleted. Each of the items in the buffer is named active element.

In this paper, we present efficient algorithms for the continuous and real-time monitoring of outliers on uncertain data streams over sliding windows. In summary, the major contributions of this work are as follows:
A new algorithm Continuous Uncertain Outlier Detection (COD) is designed for outlier detection on uncertain data streams. The algorithm is able to quickly determine the nature of an uncertain element by probabilistic pruning, to further improve the efficiency.

II. RELATED WORK

Outlier detection has been studied in the literature, both in the context of multi-dimensional data sets and in the more general case of metric spaces. Usually, the proximity among objects is used to decide if an object is an outlier or not. However, specialized techniques may also be applied (e.g., projections in the case of multi-dimensional data). Apart from the fact that outliers are important in many applications, their discovery allows the data set to be "cleaned" to apply a particular model.

Clustering-Based Approaches
They always conduct clustering-based techniques on the samples of data to characterize the local data behaviour. In general, the sub-clusters contain significantly less data points than other clusters, are considered as outliers. For example, clustering techniques has been used to find anomaly in the intrusion detection domain. In the work of, the clustering techniques iterative detect outliers to multidimensional data analysis in subspace. Since clustering-based approaches are unsupervised without requiring any labelled training data, the performance of unsupervised outlier detection is limited.

Density-Based Approaches
One of the representatives of this type of approaches are local outlier factor (LOF) and variants. Based on the local density of each data instance, the LOF determines the degree of outlines, which provides suspicious ranking scores for all samples. The most important property of the LOF is the ability to estimate local data structure via density estimation. The advantage of these approaches is that they do not need to make any assumption for the generative distribution of the data. However, these approaches incur a high computational complexity in the testing phase since they have to calculate the distance between each test instance and all the other instances to compute nearest neighbors.

Learning based Approach
First, the work called uncertain-SVDD (U-SVDD) here, addresses the outlier detection only using normal data without taking the outlier/negative examples into account. Second, U-SVDD only calculates the degree of membership of an example towards the normal example and takes single membership into learning phase. However, the work in this existing address the problem of outlier detection with a few labelled negative examples and takes data with imperfect labels into account. Based on the problem, we put forward single likelihood model and bi-likelihood model to assign likelihood values to each example based on their local behaviours.

Problem Definition
In contrast to all approaches above, focus on high dimensional as well as low-dimensional data and use reverse nearest neighbors only through the distribution of k-occurrences, taking into account the inherent relationship between dimensionality, neighborhood size and reverse neighbors that was not observed in previous outlier-detection work.
Sliding window semantics can be either time-based or count-based. In time-based window scenarios, the window size W and the Slide are both time intervals. Each window has a starting time Tstart and an ending time Tend = Tstart + W. The window slide is triggered periodically by the system time (wall clock time), causing Tstart and Tend to increase by Slide. Each window contains a set P of n objects. In general, n varies between sliding windows reflecting the differences in arrival rates. The non-expired objects are those whose arrival Time p.arr⩾Tstart. An object expires after x slides, where x=⌈w/slide⌉p.exp is the expiration time point of p. Count based windows can be deemed as a special case of time-based ones, where the window size W is measured in data objects, n is fixed for all slides, and a slide occurs after the arrival of a certain number of objects. The proposed methods are applicable to both types of windows.

III. PROPOSED APPROACH

In data stream applications, data volumes are huge, meaning that it is not possible to keep all data memory resident. Instead, a sliding window is used, keeping a percentage of the data set in memory. The data objects maintained by the sliding window are termed active objects. When an object leaves the window, we say that the object expires, and it is deleted from the set of active objects. There are two basic types of sliding windows: (i) the count-based window which always maintains the n most recent objects and (ii) the time-based window which maintains all objects arrived the last t time instances. In both cases, the expiration time of each seen object is known. The challenge is to design efficient algorithms for outlier monitoring, considering the expiration time of objects. Another important factor of stream-based algorithms is the memory space required for auxiliary information. Storage consumption must be kept low, enabling the possible enlargement of the sliding window, to accommodate more objects.

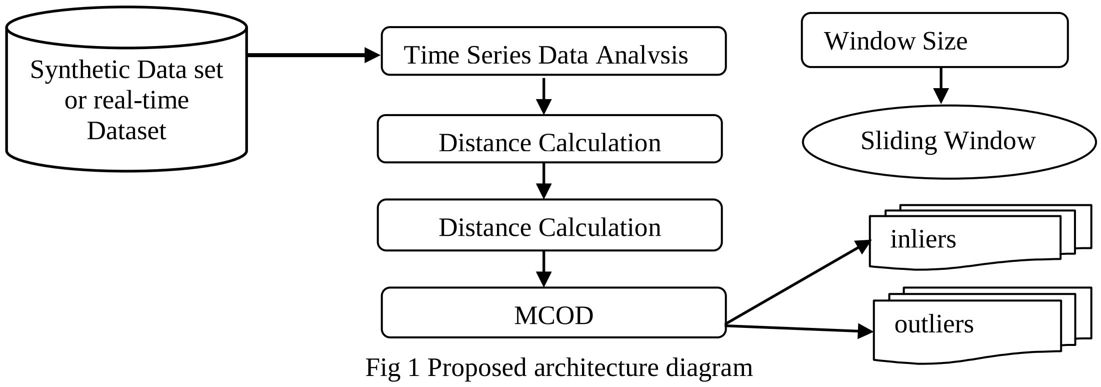

Fig 1 Proposed architecture diagram

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

A. Synthetic Data Generation

In the field of mathematical modelling, a radial basis function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Radial basis function networks have many uses, including function approximation, time series prediction, classification, and system control. The number of neurons in the hidden layer, is the center vector for neuron, and is the weight of neuron in the linear output neuron. Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name radial basis function. In the basic form all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance (although the Mahalanobis distance appears to perform better in general) and the radial basis function is commonly taken to be Gaussian. The Gaussian basis functions are local to the center vector in the sense that changing parameters of one neuron has only a small effect for input values that are far away from the center of that neuron. Given certain mild conditions on the shape of the activation function, RBF networks are universal approximates on a compact subset. This means that an RBF network with enough hidden neurons can approximate any continuous function with arbitrary precision.

B. Distance Based Model

Title The introduced system designs an efficient algorithm for continuous monitoring of distance-based outliers, in sliding windows over data streams, aiming at the clearance of the limitations of previous algorithms. The predominant concerns are surging efficiency and minimizing storage utilization. The propound algorithm is use as a basis of event-based framework that get hold of an advantage of the expiration time of objects to circumvent unwanted computations.
Distance-Based Outlier: Let S be a set of objects, obj an object of S, k a positive integer, and R a positive real number. Then, obj is a distance-based outlier (or, simply, an outlier) if less than k objects in S lie within distance R from obj.
Data Stream Outlier Query: Given a data stream DS, a window size W, and parameters R and k, the Data Stream Outlier Query is: return the distance-based outliers in the current window.

C. Event Based Approach

The system interested in tracking the outliers in a set of objects of a stream defined by a sliding window. In particular, a set of outliers is maintained subject to arrivals of new objects from the stream and departures of existing objects due to the restricted window size (either restricted with respect to time or with respect to number of objects). The arrival and departure of objects has the effect of a continuously evolving set of outliers.
An event is the process of checking whether an inlier becomes an outlier due to departure of objects from the window. The expiration time of the objects is known whether we talk about time-based windows (in this case a new object p has expiration time now + d W Slide e) or for count-based windows (in this case p expires after a predefined number of new objects have arrived). Thus, the time stamp of an event depends on the expiration time of objects. This forces a total order on the events which can be organized in an event queue. An event queue is a data structure that supports efficiently the following operations:

- Findmin: returns the event with the most recent time stamp (the most recent event).
- Extractmin: invokes a call to findmin and deletes this event from the event queue.
- Increasetime(p, t): increases the time stamp of the event associated to object p by t. It is assumed that we are provided with a pointer to p and there is no need to search for it.
- Insert (p, t): inserts an event for object p into the queue with time stamp t.

The event-based method for outliers employs an event queue to efficiently schedule the necessary checks that have to be made when objects depart. Thus, in the event queue there are only stored inliers since only these can be affected by the departure of an object. Arrival of new objects results in potential updates of the keys of some objects in the event queue. Additionally, existing outliers are checked as to whether they have become inliers and thus they should be inserted in the event queue.

D. Sliding window

The outliers in a set of objects of a stream defined by a sliding window. In particular, a set of outliers is maintained subject to arrivals of new objects from the stream and departures of existing objects due to the restricted window size (either restricted with respect to time or with respect to number of objects). The arrival and departure of objects has the effect of a continuously evolving set of outliers. At only certain discrete moments, however, this set may change and an inlier becomes an outlier or vice-versa. Between these discrete moments, the set of outliers remains as is. The idea is to focus on the temporal and geometric relations between objects to guarantee the correctness of the set of outliers for a period of time.
The effect of arrivals of objects is to turn existing outliers into inliers. On the other hand, the potential effect of departures is to turn inliers into outliers. However, the exact time of the departure of each object is pre-specified (due to the sliding window) and thus we can plan in the future the exact moments in which one needs to check whether an inlier has turned into outlier.

E. Micro Continuous outlier detection

Propose a methodology to mitigate this. Our methodology is based on the concept of evolving micro-clusters that correspond to regions containing inliers exclusively. The resulting algorithm is denoted as COD (Continuous Outlier Detection). In a more complex scenario, multiple users could be interested in the distance-based outliers over a data stream. However, each user comprehends the notion of outlier differently by varying values of R and k. Each pair of R and k determines a query q of distance-based outlier detection. Therefore $\mathcal{D}(q.R\ ,\ q.k)$denotes the outliers of query q from the set of all queries Q. In this section, we study the continuous evaluation of multiple queries. For simplicity, we discuss separately the case in which k varies and R remains constant and vice-versa. At the end, we combine trivially both methods into one so that both parameters can vary. The algorithms are similar to the ones discussed in the previous section (both variations). Here we only report the changes. We continuously evaluate the query with the maximum value of parameter k. When an object departs, if the examination of an object p, at p.ev time instance, reports p as outlier we check the other queries in Q whether p is also outlier in them. In particular, for each query q, if$n_p^- + n_p^+ < q.k$, then p is outlier in q.
Queries are examined with decreasing order of k, and this procedure is terminated as soon as we reach a query for which p is inlier. Moreover, when a new object arrives, if object$p \in \mathcal{D}(R, q.k_{max}$ and its counter n_p^+ is increased, we check all the queries for a possible move of p from outlier set to inlier set. Notice that p is not necessarily outlier in all queries. For each query q, if $p \in \mathcal{D}(R, q.k)\ \ and\ n_p^- + n_p^+ > q.k$then p should be removed from $\mathcal{D}$(R, q.k). The queries are examined again in decreasing order of k and the procedure is stopped when we reach a query in which p is not outlier.
Proceed now with the examination of the case of fixed k and varying R. In this case, two sets for each object p are maintained, the sets $P_p$and S_p(recall that we only stored the size of S_p) along with their distances from p, by taking into account the maximum distance $R_{max}\ = \max\{q_i, R\}(0 \le i \le |Q|$When R varies it is necessary to maintain S_p since the neighbors of an object depend on the radius of the query.
This may lead to high memory requirements, since in the worst case the number of neighbors can reach the number of active objects n. In the sequel, we study a more efficient scheme in terms of memory requirements. Assuming the maximum distance $R_{max}\,, if\ n_p^+ > k$ we can maintain the k neighbors with the smaller distances from p. This is because neighbors with larger distances will not be used in any query. Therefore, the size of S_p is limited to k objects.
The key idea is the observation that all the preceding neighbors of p, which may have an impact on whether p is outlier or not, belong to the answer of the k-1skyband query in the expiration time - distance space. A K'-skyband query reports all the objects that are dominated by at most k0 other objects. Therefore K'-skyband equals to the skyline query. In our case, the maximization of the expiration time and the minimization of the distance determine the domination relationship between objects, i.e., an object dominates another object if it has greater expiration time and smaller distance from p. The rationale of this observation is that at each time instance, the k nearest objects top belong to the K'- sky band of the preceding neighbors. The main rationale behind our approach is to drastically reduce the number of objects that are considered during the range queries when these are performed. The detailed steps of the modified algorithm after each window slide are as follows:

Step 1: The expired objects are purged after having updated the counters mcn of corresponding micro-clusters (if any), accordingly. Subsequently, steps 2 and 3 are performed for each new data object p; new objects are processed in the order of their arrival.

Step 2: For each p, we detect (i) the micro-cluster, the center of which is closest to that object, and (ii) all micro-clusters, the centers of which are within a 32 R range. Conflicts (i.e., when there are two centers with equal distance) are resolved arbitrarily. Note that we can employ a specific structure to store the micro-cluster centers, such as an M-tree, to perform this task efficiently.

Step 3: If the distance from the closest center is not greater than R/2, then: (3a-i) the new object is assigned to the corresponding micro cluster and the value of p.mc is updated; (3a-ii) the size of the corresponding micro-cluster is increased by one; (3a-iii) let MCi be the micro-cluster where the new object is inserted. We evaluate the distance between the new object and all objects in PD that contain MCi in their Rmc lists, to check (i) whether the number of succeeding neighbors of the latter should be increased and (ii) whether any previous reported outliers have become inliers; Otherwise, i.e., if the distance from the closest center is greater than R/2, no assignment takes place and the following process is applied: (3b-i) For the new object p that has not been assigned to a micro-cluster, we perform a range query taking into account only (i) the objects in PD and (ii) the objects in the microclusters for which the distance from their centers is not greater than 32 R (the relevant micro-clusters have been detected in Step 2). (3b-ii) If the number of neighbors from the PD set within R/2 distance exceeds μk, μ Π 12, then a new micro-cluster is created, with the new object as its center. All the corresponding objects are moved from PD to Imc. All objects still in PD that are less than 32 R apart update their Rmc lists with the identifier of the new micro-cluster. (3b-iii) Otherwise, the event (i.e., creation of the list of the expiration times of the neighbors of the new object and update of the number of succeeding neighbors) is applied. The objects in p.Rmc are the cluster identifiers for which the distance from their centers is not greater than 32 R.

Step 4: If the size of a micro-cluster shrinks below k + 1, then this micro-cluster is dissolved, and its former objects are treated in a way similar to that described in Step 3b. At the end of these steps, additional outliers are reported with the help of the event queue, which in MCOD, does not include any object p 2 Imc. The main advantage compared to the algorithms in the previous sections is that the number of distance computations is reduced significantly.

IV. EXPERIMENTAL RESULTS

We have conducted a series of experiments to evaluate the performance of the proposed algorithms. We compare algorithms MST, LOF, HD Outlier and MCOD against the algorithm, which is termed sliding window. Which requires k and R to be fixed, since its functionality is covered by MCOD algorithm. All methods have been implemented in java and the experiments have been conducted on a Pentium@3.0GHz Win7 machine with 4GB of RAM. The performance of outlier detection algorithms can be evaluated based on two error rates: detection rate and false alarm rate.

A. outlier detection accuracy

Detection rate gives information about the number of correctly identified outliers, while the false alarm rate reports the number of outliers misclassified as normal data records. The detection rate and the false alarm are computed as follows: Detection rate = TP/TP + FN, False alarm rate =FP/FP + TN. The precision and recall measures were employed. The precision represents the fraction of objects reported by the algorithm as outliers that are true outliers. The recall represents the fraction of true outliers correctly identified by the algorithm's.

TABLE I OUTLIER DETECTION ACCURACY WITH DIFFERENT ALGORITHM

Dataset	MST	LOF	Hd outliers	MCOD
Airlines	0:17	0:12	0:25	0:47
Cov type Norm	0:19	0:15	0:35	0:71
kddcup99	0:20	0:14	0:36	0:69
Reuters	0:26	0:16	0:51	0:94

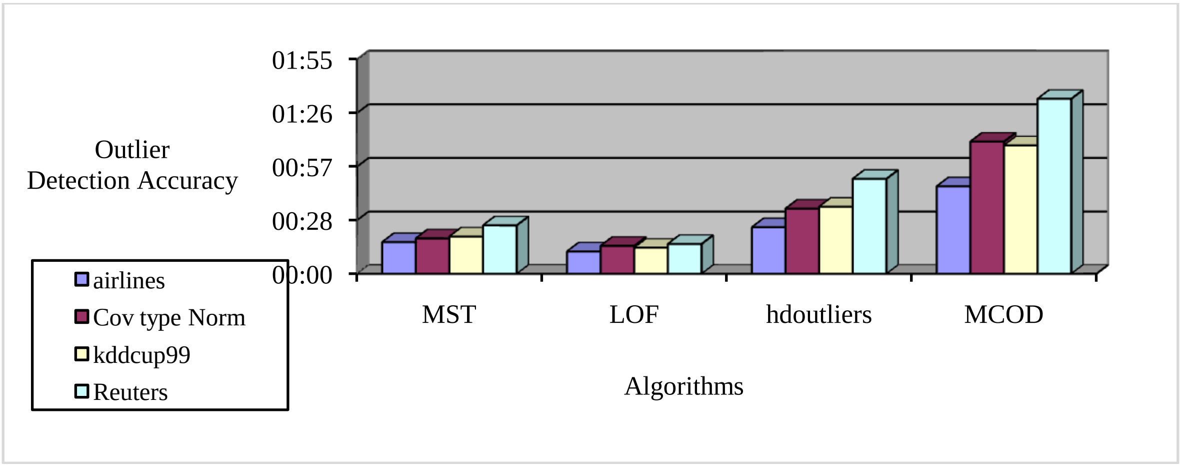

Fig 2. Comparison of Outlier Detection accuracy

B. CPU Times

First, we study the performance of the methods for varying values of W in the range [10K; 1000K]. Depicts the results. For Slide = 1, the memory requirements for Abstract- C are very high. More specifically, Abstract-C Stores W¢(W¡1) 2 counters, which corresponds to 74GB for W = 200K and 465GB for W = 500, assuming integers need 4 bytes. Because of that, in Figure 3, Slide = 1 only for STROM, LOF, HDoutlier and MCOD, while we choose Slide = 0:001W for Abstract- C. Despite that favourable configuration, Abstract-C performs significantly worse than our algorithms in terms of running time.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

TABLE II COMPARISON DIFFERENT WINDOWS SIZE VS CPU TIME

Window Size (K)	**MST**	**LOF**	**Hd outliers**	**MCOD**
10	10	0.8	0.7	0.1
100	11	0.9	0.74	0.3
200	12.5	0.95	0.8	0.4
300	12.8	1.5	1	0.8
400	13	1.8	1.3	1
500	13.5	1.9	1.45	1.1

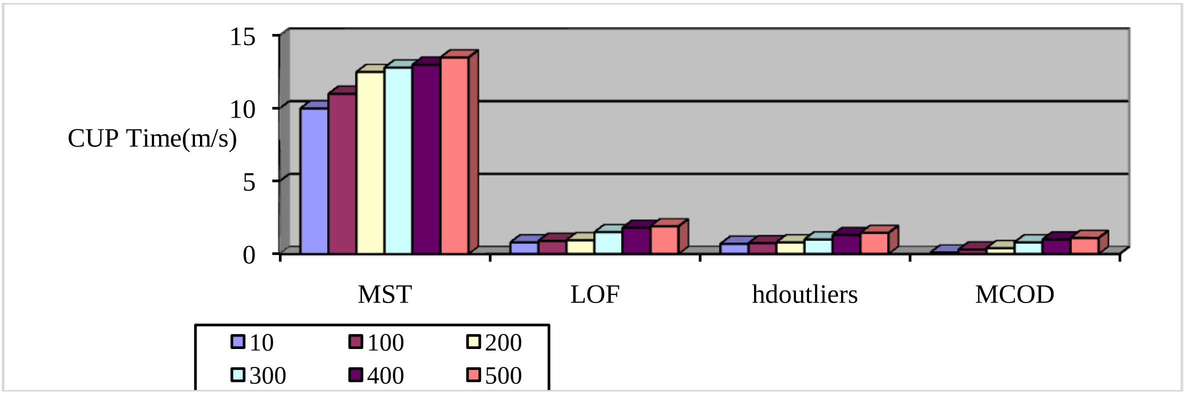

Fig 3. CPU Time for outlier detection

C. Outliers % (W)

The memory requirements, the number of distance computations and other qualitative measurements. Sliding windows have been used, whereas time-based ones are supported, without significant changes in the results. The default values for the parameters (unless explicitly specified otherwise) are: W = n = 200K, jQj = 1, i.e., there is a single query, k = 10 and the parameter R is set in a way that the number of outliers jDj = (0:01 0:001) n. Since we want to investigate the most demanding form of continuous queries, we set Slide = 1. All measurements correspond to 1000 slides, i.e., 1000 insertions/deletions in P.

TABLE III COMPARISON DIFFERENT PERCENTAGE OF OUTLIER DETECTION

outliers (% W)	**MST**	**LOF**	**Hd outliers**	**MCOD**
0.1	**0.8**	**0.8**	**1**	**2**
0.5	**1**	**1.5**	**1.2**	**2.5**
1	4	3	3.4	4.5
1.5	5	4.6	3.6	5.3
2	8	6.4	5.3	6.65
2.5	9	7.4	6.3	8.4
3	10	8.5	7.8	9.8

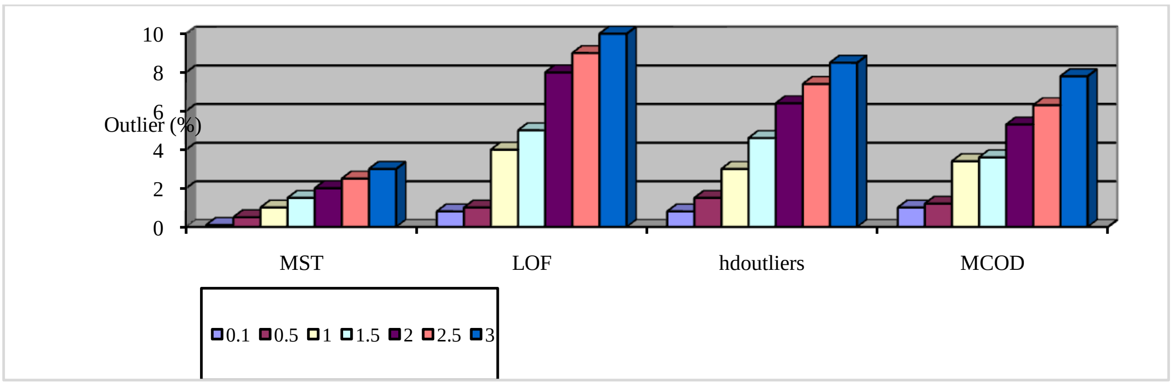

Fig 4. Percentage of outlier detection

D. Memory Consumption

The memory consumption of the two real data sets for the experiment. The consumed memory corresponds to the memory needed to store the information for each active object (i.e., preceding and succeeding neighbors), the heap size used for the events prioritization.

TABLE IV MEMORY USAGE VS DIFFERENT ALGORITHM

W	**MST**	**LOF**	**hdoutliers**	**MCOD**
10,000	0.48	2.95	4.29	0.27
200,000	9.60	100.45	111.85	4.94
300,000	14.47	133.27	194.28	11.04
400,000	19.32	178.30	280.37	15.40
500,000	24.23	232.72	377.51	20.23

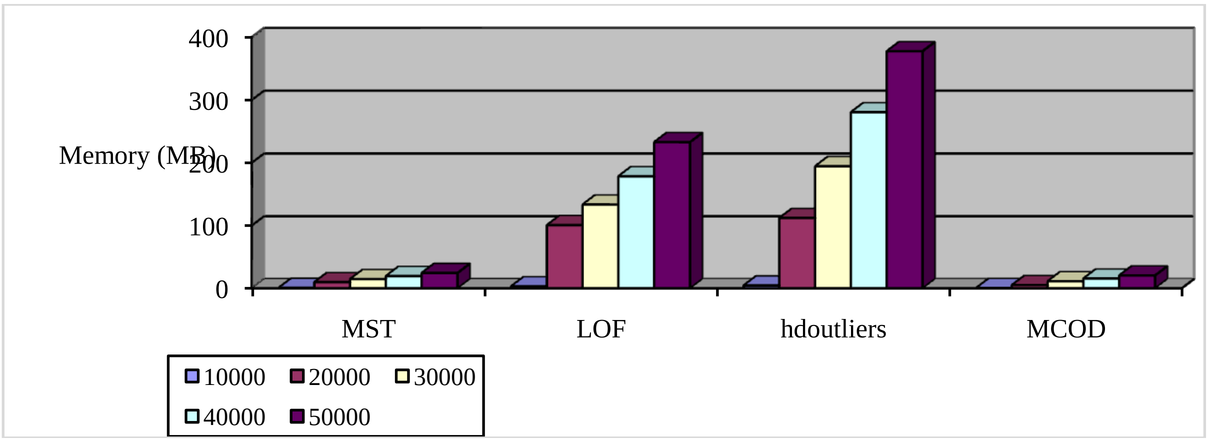

Fig 5. Comparison of memory consumption for different algorithms

The outliers of all the queries and the micro-cluster information. As can be seen, the required amount of memory is only a small fraction of the total memory available in modern machines, even for the MCOD method

V. CONCLUSION

Anomaly detection is an important data mining task aiming at the selection of some interesting objects, called outliers that show significantly different characteristics than the rest of the data set. In this project the problem of continuous outlier detection over data streams, by using sliding windows. More specifically, COD algorithms are design, aiming at efficient outlier monitoring with reduced storage requirements. This method doesn't make any assumptions regarding the nature of the data, excepts from the fact that objects are assumed to live in a metric space. As it is shown in the performance evaluation results, based on real-life and synthetic data sets, the proposed techniques are by factors more efficient than previously proposed algorithms. An interesting direction for future work is the design of randomized algorithms for detection, aiming at significant improvement of efficiency by sacrificing the accuracy of results.

REFERENCES

[1] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in SIGMOD Conference, 2000, pp. 427–438.
[2] E. Knorr and R. Ng, "Algorithms for mining distance-based outliers in large data sets," in VLDB Conference, 1998.
[3] E. Knorr, R. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," The VLDB Journal, vol. 8, no. 3-4, pp. 237–253, 2000.
[4] D. Yang, E. Rundensteiner, and M. Ward, "Neighbor-based pattern detection for windows over streaming data," in EDBT, 2009, pp. 529– 540.
[5] Y. Zhu and D. Shasha, "Statstream: statistical monitoring of thousands of data streams in real time," in VLDB Conference, 2002, pp. 358–369.
[6] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in VLDB Conference, 1997, pp. 426–435.
[7] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in SDM, 2006.
[8] Aggarwal C C. On density-based transforms for uncertain data mining. In Proc. the 23rd International Conference on Data Engineering, April 2007, pp.866-875.
[9] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in Proc. ACM SIGMOD Int. Conf. Manage. Data, New York, NY, USA, 2000, pp. 93–104.
[10] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori, "Statistical outlier detection using direct density ratio estimation," Knowl. Inform. Syst., vol. 26, no. 2, pp. 309–336, 2011.
[11] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," IEEE Trans. Knowl. Data Eng., vol. 21, no. 5, pp. 609–623, May 2009.
[12] Y. Shi and L. Zhang, "COID: A cluster-outlier iterative detection approach to multi-dimensional data analysis," Knowl. Inform. Syst., vol. 28, no. 3, pp. 709–733, 2011.