A Data Stream Outlier Delection Algorithm Based On Reverse K Nearest Neighbors

Cao Lijun [1] , Liu Xiyin [2] , Zhou Tiejun [1] , Zhang Zhongping [3], Liu Aiyong [1]
1Hebei Normal University of Science & Technology , Qinhuangdao,Hebei, 066004, P.R.China
[2]Yaohua Design Institute ,Qinhuangdao, Hebei, China
[3]College of Information Science and Engineering Yanshan University, Qinhuangdao,Hebei, 066004, P.R.China
E-MAIL:Misscao6666@163.com, liuxiyin2003@sina.com, zamjam99@yahoo.cn

***Abstract*—A new data stream outlier detection algorithm SODRNN is proposed based on reverse nearest neighbors. We deal with the sliding window model, where outlier queries are performed in order to detect anomalies in the current window.. The update of insertion or deletion only needs one scan of the current window, which improves efficiency. The capability of queries at arbitrary time on the whole current window is achieved by Query Manager procedure, which can capture the phenomenon of concept drift of data stream in time. Results of experiments conducted on both synthetic and real data sets show that SODRNN algorithm is both effective and efficient.**

Keywords Data stream;Outlier; Reverse k nearest neighbors; Sliding window

I. INTRODUCTION

In many emerging applications, such as fraud detection, network flow monitoring, sensor network data management, video tracking[1],etc., data arrive continuously and unboundedly, but it is either unnecessary or impractical to store all incoming objects. In this context, an important challenge is to find the most exceptional objects (called outliers) in the data stream. Outlier detection is a key issue in traditional data mining. There are already several kinds of outlier detection algorithms, such as statistic-based methods, distance-based methods, density-based methods and deviation-based methods, etc. Traditional methods need to perform multiple scans of the database, they are not suitable for the data stream environment. It is very important to detect data stream outliers through only one scan of the stream.

Outlier detection in data streams has attracted more and more attention from researchers. A distributed framework to approximate data distributions coming from a sensor network is presented In [2]. It is used to report outlying values in the union of the readings coming from multiple sensors. This algorithm has good efficiency, but is specifically designed to support sensor networks. In [3], a data stream outliers detection algorithm based on k-means partitioning DSOKP is proposed，which applies k means clustering on each partition of the data stream to generate mean reference point set, and subsequently picks out those potential outliers of each periods according to the definition of outliers. DSOKP is better in efficiency, but it can not capture the concept drift information.

An incremental outlier mining algorithm based on LOF[4] is presented in both [5] and [6]. It only updates the local outlier factor of the influenced objects when new object is inserted or expired object is deleted. This algorithm can be used in stream environment, but it demands three scans of the data set to update the k distance and local reachable density and local outlier factor of the influenced objects. In order to avoid multi-scan of the data set and to capture concept drift, this work proposes a novel stream outlier detection algorithm based on reverse k nearest neighbors. The update of the current window because of insertion and deletion needs only one scan.

II. PROBLEM DEFINITIONS

Definition 1: (Data Stream) A data stream DS is a possible infinite series of objects

{…at-N+1,at-N+2,… ,at-1,at,…}

where at denotes the object observed at time t, N is the width of the current window. In the sliding window model, the window is identified by two sliding endpoints at-N+1 and at.

Definition 2: (Hawkins's definition to outlier)[7] An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism. Hawkins's definition is intuitive and it captures the general spirit of an outlier.

Definition 3: (k-distance of an object p) For any positive integer k, the k-distance of object p, denoted as k-distance(p), is defined as the distance d(p,o) between p and an object $o \in D$ such that:

(1) for at least k objects $o' \in D\backslash\{p\}$ it holds that $d(p,o') \le d(p,o)$

(2) for at most k-1 objects $o' \in D\backslash\{p\}$ it holds that $d(p,o') < d(p,o)$

Definition 4: (k-distance neighborhood of an object p) Given the k-distance of p, the k-distance neighborhood of p contains every object whose distance from p is not greater than the k-distance, i.e.

Nk-distance(p)(p)=$\{q \in D\{p\} | d(p, q) \le$ k-distance(p) $\}$

These objects q are called the k-nearest neighbors of p.

Definition 5: (reverse k nearest neighborhood of p) The reverse k-nearest neighborhood RNN is an inverse relation which can be defined as:

RNNk(p) $=\{q|q \in D, p \in$ Nk(q)$\}$. $|$RNNk(p)$|$

denotes the number of the reverse k-nearest neighbors of object p.

Definition 6: (rknn-outlier) The rknn outlier is the data object in the current window which has the least number of reverse k nearest neighbors. If object p has less reverse k nearest neighbors in the current window,from the definition of RNN, we know that it is because less objects regard object p as their k nearest neighbors. So the outlierness of object p is much higher than that of others. The rknn-outlier conforms well to the essence of Hawkins's outlier definition.

An important character of data stream is concept drift. Due to data stream evolution, object properties can change over time and, hence, evaluating an object for outlierness when it arrives, although meaningful, can be reductive in some contexts and sometimes misleading. On the contrary, by classifying single objects when a data analysis is required, data concept drift typical of streams can be captured. To this aim, it is needed to support queries at arbitrary points-in-time, called query times, which classify the whole population in the current window instead of the single incoming data stream object.

III. Algorithm

A. Algorithm method

In this section the algorithm SODRNN, standing for Stream Outlier Detection based on Reverse k Nearest Neighbors, is described. This algorithm consists of two procedures: the Stream Manager and the Query Manager. And the entire window should be allocated in memory.

The former procedure receives the incoming data stream objects and efficiently updates the current window. When new stream object comes, in order to maintain current window perfectly, it needs only update the knnlist and rknnlist of the influenced objects in the current window rather than that of all the data stream objects in the current window. When new coming object is inserted, it needs only one pass of scan to the current window to find all objects whose k nearest neighbors are influenced. The update of the knnlists of the influenced objects in the current window can update their rknnlists at the same time. The deletion of the expired object needs only update the rknnlists of the influenced objects in the current window according to its knnlist, and then update the knnlists of the influenced objects in the current window according to its rknnlist.

When user demands a query of the top m outliers, the latter procedure will make a scan of the current window and return m objects whose $|RNNk(p)|$ is small as outliers of this query.

A node n is a record consisting of the following information: n.obj, a data stream object; n.id, the identifier of n.obj; n.knnlist[],a list, containing the identifiers and the distances of the most recent k nearest neighbors of n.obj, it is ordered by distance; n.rknnlist[],a list, having size at most k, containing the identifiers of the most recent reverse k nearest neighbors of n.obj, this list is ordered by id. We assume that both the operation of ordered insertion of a novel identifier in the list and the operation of search of an identifier in the list are executed in time O(log k).

B. Algorithm description

```
ALGRITHM: SODRNN
INPUT:    DS, current window size N, integer k, query time Uquery, number of outlier m
OUTPUT:   m outliers
METHOD:
BEGIN
(1) SM(DS,N,k);
(2) when (Uquery) QM(m);
END
```

C. Stream Manager Procedure

```
PROCEDURE SM(DS,N,k)
BEGIN
(1) FOR each data stream object obj with arrival time t DO
(2)  IF the oldest object q of current window expires
(3)FOR all objects o in q.knnlist DO   o.rknnlistdelete(q);
(4)          FOR all objects o in q.rknnlist DO o.knnlistdelete(q);
(5)  ENDIF
(6)  remove object q from current window
(7)   object p(obj,t,Φ,Φ);         //create a new node p for obj with p.obj = obj, p.id = t
(8)  FOR all objects o in current window DO
//update current window
(9)    dist=o.distance(p);
(10) p.knnlistinsert(o); //the k nearest neighbors of p
(11)   o.rknnlistinsert(p);
(12)   IF dist<=o.k_distance()
(13)     o.knnlistinsert(p);
(14)         p.rknnlistinsert(o); //the reverse k nearest neighbors of p
(15)   ENDIF
(16)  ENDFOR
(17)  Insert object p into current window.
(18) ENDFOR
END
```

```
Outlier Query Management Procedure
PROCEDURE QM(m)
BEGIN
(1) perform a single scan of current window;
(2) return m objects with minimal |RNNk(p)| as outliers.
END
```

Spatial Analysis

In the following we use N to denote the size of the current window. Each node consists of four members: n.obj, n.id, n.knnlist[], n.rknnlist[], we use d to denote the space required to store an object n.obj, an integer which stores the identifier of n.obj. For all objects of the current window, when there is a k nearest neighbor, there must be a reverse k nearest neighbor correspondingly somewhere in the current window due to the definition of the reverse k nearest neighbor. So for the whole current window the total number of the k nearest neighbors and that of the reverse k nearest neighbors are equal to Nk. Therefore, the spatial cost of the SODRRN algorithm is O(N(d+2k+1)).

Temporal analysis

The stream manager procedure consists of three main stages when processing a new object. Firstly, update current window and delete the expired node q. Step(3) updates rknnlists of the objects influenced by the deletion, whose cost is O(klogk); Step(4) updates knnlists of the objects influenced by the deletion, whose cost is also O(klogk). This is because we can find directly the objects whose k nearest neighborhood are influenced. We need not make a whole scan of the current window by using the rknnlist. It is noticeable that when the distance between q and o is less than or equal to o.k_distance() and o.knnlist has only k objects we should take a knn query to find the nearest object of o and insert it into o.knnlist, whose cost is O(klogklogN). Step(6) has constant cost. So the first main stage has a cost of O(klogklogN). When k is far less than N, the cost is O(logN). Secondly, the creation of new node p has constant cost. Thirdly, update the current window and insert node p into it. Step(8) finds all the objects whose k nearest neighborhood are changed, it takes a cost of O(N), Step(9) has constant cost.; Step(10) has a cost of O(logk); Step(11) updates rknnlists of the objects influenced by the insertion, whose cost is O(logk); Step(13) updates the knnlists of the objects influenced by the insertion when the distance of p and o is less than o.k_distance(), whose cost is O(logk); It is noticeable that if there already exist k-1 objects within o.k_distance() we should delete the farthermost nearest neighbor r of object o and update r.rknnlist. In this case, the insertion of p into o.knnlist has a cost of O(log2k); Step(14) has a cost of O(logk); Step(17) has constant cost. So the third stage has a cost of O(Nlog2k), When k is far less than N, the cost is O(N). Summarizing, the cost of the stream manager procedure is O(N).

The query manager procedure has a time cost of O(Nlogm), due to m<<N, procedure QM has a cost of O(N).

IV. EXPERIMENTAL RESULTS

Our experiments were performed on several synthetic data and real data sets. In all our experiments, we have assumed that we have information about the outliers in the data set, so we could evaluate the detection performance.

In order to improve the knn query performance, we also make a change to the X-tree index structure. We omit its SplitHistory information field of the node structure and use a simple clustering algorithm for the splitnode algorithm which supports knn query better. Then we implement the program SODRNN based on this index structure with VC++6.0. The experiments are conducted on an Intel Pentium D 3.1GHz PC with 1GB main memory under Windows XP.

A. *Experiments on real data sets*

In this paper, in order to verify the validity of the proposed algorithm, we employed the KDD Cup 1999 benchmark evaluation data sets, which has been extensively used to evaluate intrusion detection algorithms. The data consists of network connection records of several intrusions simulated in a military network environment. As the original data sets is too large, we used only 50,000 TCP connection records from about one week of data. Each record of the TCP connections has been elaborated to construct a data set of 23 numerical features, which consists of duration, src_bytes, num_file_creations, num_failed_logins, etc. Intrusion behaviors account for 1% of the entire data set, which consist of 4 types of intrusion behaviors: DOS(denial of service),R2L(unauthorized access from a remote machine), U2R(unauthorized access to local super user privileges), probing(surveillance and other probing).

The TP(True Positive rate) and FP(False Positive rate) measures were employed. TP represents the fraction of objects reported by the algorithm as outliers that are true outliers. FP represents the fraction of true outliers incorrectly identified by the algorithm. An outlier query was submitted every one hundred objects, and the first query was submitted only after having observed the first N data stream objects. Results reported in the sequel are averaged over the total number of queries submitted. In order to study the influence of the window size to the detection result, we chose several window width to perform experiments respectively. The results are shown in figure 1, where k has the value of 10. As can be seen form figure 1, the TP and FP of our algorithm is 85.78% and 8.83% respectively when the window size N is 2,000. With the increasing of the window size, TP is up and FP is down. This is because when a new object is inserted it is only influenced by its preceding N objects and when it becomes expired it will only influence its succeeding N objects. Therefore, the influence domain of an object is 2N. With the increasing of N, the value of $|RNNk(p)|$ becomes more accurate. Data stream has the character of concept drift, however, the detection rate can only reach 100% when N is $+\infty$. To store all incoming objects is either unnecessary or impractical. As our experiments show, the TP and FP has reached 99.44% and 1.77% respectively when N is 10,000. The experiment confirms our algorithm is effective.

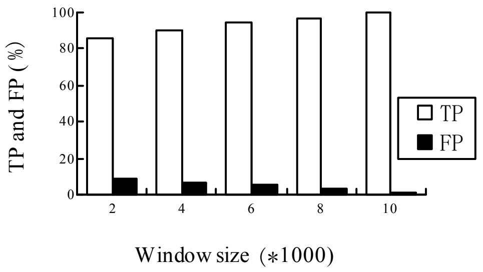

Figure1. The effectiveness of SODRNN (kdd cup99)

B. *Experiments on synthetic data sets*

In order to test the efficiency of our algorithm, we used a two dimensions synthetic data set with 30,000 records which has three main clusters. We inserted one percent

records as outliers that has certain difference with the three main clusters. The window size N was set to 10,000. We performed experiments to SODRNN and IncLOF algorithm respectively. In this experiment, different values of k were chosen when comparing the time cost of processing a single record. As we can see from figure 2, the processing time of the two algorithm both are increased nearly linearly with the increasing of k.. The increasing speed of SODRNN is slower than that of IncLOF. The processing time of

SODRNN is much less than that of IncLOF when k is the same value. This experiment confirms that the processing time of SODRRN is linear with the value of k and SODRNN is more efficient than IncLOF. When k is in the same value, the time cost of SODRNN is an order of magnitude lower than that of IncLOF.

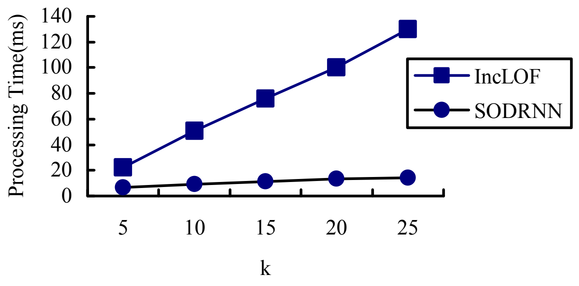

FIGURE2. THE PROCESSING TIME OF SINGLE OBJECT AT DIFFERENT

V. CONCLUSIONS

In this paper, a novel data stream outlier detection algorithm SODRNN is presented. In contrast to IncLOF, this algorithm reduces the number of scans to only one. Experiments conducted on both synthetic and real data sets show that the proposed method is efficient and effective. The future work is to improve the performance of this algorithm in high-dimensional data stream environment.

REFERENCES

[1] [1] Han J, Kamber M, Data Mining: concepts and techniques, 2nd edn. Morgan Kaufmann,2006:1-29

[2] [2] Song Xiuyao, Wu Mingxi, Jermaine C. Statistical change detection for multi-dimensional data.KDD,California,2007:667-676

[3] [3] Wu Mingxi, Jermaine C.A Bayesian method for guessing the extreme values in a data set.In Proc of VLDB,Vienna,Austria,2007:471-482

[4] [4] Abe N, Zadrozny B, Langford J. Outlier detection by active learning. Pennsylvania, KDD,2006:504-509

[5] [5] Barbara D, Domeniconi C, Rogers J P. Detecting outliers using transduction and statistical testing. Pennsylvania,KDD,2006:55-64

[6] [6] Aggarwal C C, Yu P S. An effective and efficient algorithm for high-dimensional outlier detection. The VLDB Journal,2005:211-221

[7] [7] Lazarevic A,Kumar V.Feature bagging for outlier detection.Chicago,KDD, 2005:157-166

[8] [8] Y. Dora Cai, David Clutter, Greg Pape, Jiawei Han. MAIDS: Mining alarming incidents from data streams SIGMOD,Paris,2004:919-920

[9] [9] Palpanas T, Papadopoulos D, Kalogeraki V, et a1. Distributed deviation detection in sensor networks.SIGMOD Record,2003,32(4):77-82

[10] [10] Subramaniam S,Palpanas T.Online outlier detection in sensor data using non-parametric models. VLDB,Seoul,2006:187-198

[11] [11] Z Musa, J Watada,Video tracking system: a survey, *ICIC EXPRESS LETTERS,* vol.2,no.1,pp.65-72,2008.

[12] [12] S Subramaniam, T Palpanas, D Papadopoulos, et al. Online outlier detection in sensor data using non-parametric models. *Proc. of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea, pp.187-198, 2006.

[13] [13] Weiwei Ni, Jieping Lu, Geng Chen, et al. An Efficient Data Stream Outliers Detection Algorithm Based on k-Means Partitioning. J. of Computer Research and Development, vol.43,no.9,pp.1639-1643,2006(in Chinese).

[14] [14] M M Breuning, H P Kriegel, R T Ng, et a1.LOF: Identifying density-based local outliers. In Proc. of SIGMOD,Texas ,pp. 93-104,2000.

[15] [15] Fengzhao Yang, Yangyong Zhu, Baile Shi. IncLOF: An Incremental Algorithm for Mining Local Outliers in Dynamic Environment. J. of Computer Research and Development, vol.41,no.3,pp. 477-484,2004(in Chinese).

[16] [16] D Pokrajac, A Lazarevic .Incremental local outlier detection for data streams. Proc. of the 2007 IEEE Symposium on Computational Intelligence and Data Mining, Honolulu, Hawaii, pp. 504-515,2007.

[17] [17] D Hawkins. Identification of outliers. Chapman and Hall, London,1980.

[18] [18] Pokrajac D, Lazarevic A .Incremental local outlier detection for data streams. IEEE CIDM,2007:504-515

[19] [19] Muthukrishnan S, Shah Rahul, Vitter Jeffrey Scott. Mining deviants in time series data streams. 16th International Conference on Scientific and Statistical Database Management, SSDBM 2004:41-50

[20] [20]Angiulli F, Fassetti F. Detecting distance-based outliers in streams of data. Lisbon, Portugal, November 6-10,2007,CIKM 2007:811-820