


ITC 1/48 Journal of Information Technology and Control Vol. 48 / No. 1 / 2019 pp. 34-46 DOI 10.5755/j01.itc.48.1.21162	An Efficient Outlier Detection Approach Over Uncertain Data Stream Based on Frequent Itemset Mining	
	Received 2018/07/06	Accepted after revision 2018/12/18
	 http://dx.doi.org/10.5755/j01.itc.48.1.21162	

An Efficient Outlier Detection Approach Over Uncertain Data Stream Based on Frequent Itemset Mining

Shangbo Hao, Saihua Cai, Ruizhi Sun, Sicong Li

College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China;
 e-mails: hao_caumail@cau.edu.cn, caisaih@cau.edu.cn, sunruizhi@cau.edu.cn, lsc@cau.edu.cn

Corresponding author: sunruizhi@cau.edu.cn

Outlier detection is essential in data-based science. It aims to detect those itemsets that have a significant difference from the other data. With the limitations of equipment precision and network transmission, uncertain data are becoming more common in daily life. However, the traditional outlier detection methods are not applicable for uncertain data stream, and the large volume of data makes outlier detection costly in terms of memory usage and time. Moreover, the multiple scanning of the data stream required for Apriori-like methods is unrealistic. In this paper, a matrix structure is constructed to store the information of an uncertain data stream, and the subsequent mining process is conducted on the matrix structure; therefore, the whole data stream needs to be scanned only once. Then, the “*upper cap*” concept is used in the FIM-UDS method to mine the frequent itemsets more effectively to support outlier detection. Moreover, two outlier factors and an outlier detection method called FIM-UDSOD are designed to detect potential outliers. Finally, two public datasets are used to verify the efficiency of the FIM-UDS method, and one synthetic dataset is used to evaluate the FIM-UDSOD method. The experimental results show that our proposed FIM-UDSOD method is more effective than other methods in detecting outliers.

KEYWORDS: outlier detection, frequent itemset mining, uncertain data stream, outlier factors.

1. Introduction

As a main form of data, data stream is increasingly common in daily life due to the wide use of sensors. However, outliers (abnormal data) exist with the emergence of data

stream, and they have seriously affected the accuracy of the data. Therefore, the outliers need to be detected as soon as possible to allow better use of the collected data.

The traditional outlier detection methods can be divided into clustering-based methods [9, 14], distance-based methods [1, 2, 3, 10], KNN-based (K-nearest neighbor) methods [13], density-based methods [15, 16] and frequent itemset mining-based methods [4, 7]. The traditional clustering-based outlier detection methods, distance-based outlier detection methods, KNN-based outlier detection methods and density-based outlier detection methods can detect implicit outliers. However, the increasing frequency of subitemsets has not been taken into consideration in outlier detection; therefore, the detected outliers are not consistent with the definition of outliers proposed by Hawkins [6]. Aimed at this problem, the itemset mining-based outlier detection methods fully consider the influence of the frequency of the existing itemsets on abnormality judgment. However, the outlier judgment condition of the existing itemset mining-based outlier detection method, FindF-POF [7], is very simple, which makes the efficiency of outlier detection very poor when the length of the detected transactions is not constant. Moreover, the most common outlier detection methods [7, 13-16] are directed toward static *precise* data (the data's existence or nonexistence has been determined), which are not suitable for uncertain data stream (each data element has an existential probability), although the data are only added as a probability attribute. Although window-based technologies such as sliding windows [20], damped windows [19] and landmark windows [17] provide good solutions for processing data stream, the large volume of data stream causes the frequent itemset mining processes to require considerable time cost. The multiple scanning of the data stream in methods such as some Apriori-like methods [5] and FP-growth-like methods [12, 18, 20] is also unrealistic in the era of big data.

Another problem we need to solve is the design of an evaluation index for outliers. Such an index is an important factor for outlier detection, and the benefits and drawbacks of the designed evaluation index directly determine the outlier detection efficiency. However, the evaluation index of *precise* data is not suitable for uncertain data, and no evaluation index has existed for uncertain data stream until now.

Based on the problems of outlier detection over the uncertain data stream listed above, this paper presents an efficient outlier detection method. The main contributions are summarized as follows:

- 1 The influence of the frequency of each itemset on abnormality judgment is fully considered to make the detected outliers more consistent with the definition of outliers proposed by Hawkins.
- 2 An algorithm directed into **F**requent **I**temset **M**ining over **U**ncertain **D**ata **S**tream (called FIM-UDS) is proposed. It uses a matrix to store the information existing in an uncertain data stream and then computes the “*upper cap*” before frequent itemset mining to reduce the potential scale of the extended itemsets. The longer frequent itemsets are mined with the extending process of frequent 1-itemsets.
- 3 A **F**requent **I**temset **M**ining-based **O**utlier **D**etection method over **U**ncertain **D**ata **S**tream (called FIM-UDSOD) is proposed to effectively detect the implicit outliers based on the two designed deviation factors.

The remainder of this paper is organized as follows. Related work is introduced in Section 2. The preliminaries and problem definition are presented in Section 3. The outlier detection method, including frequent itemset mining and outlier detection, is introduced in Section 4. The experimental results and discussion are presented in Section 5. Finally, the conclusions of this paper are discussed in Section 6.

2. Related Work

In this section, we introduce some related work corresponding to this paper, including (1) frequent itemset mining on data stream and (2) outlier detection.

2.1. Frequent Itemset Mining on Data Stream

In recent years, several frequent itemset mining algorithms have been proposed to mine frequent itemsets on data stream. Lim and Kang [12] proposed TwMinSwap to track recent frequent items in high-speed data stream and TwMinSwap identifies recent frequent itemsets from high-speed data stream. Aimed at the sliding window-based data stream environments, Yun and Lee [20] proposed a new algorithm called WEPS to extract the weighted erasable patterns, where the new tree and list data structures were applied in the mining process. During the pattern expanding operations, only the basic information of the tree's data was stored in the list structure,

in order to improve mining efficiency. To mine more meaningful patterns, the gain values of patterns and different item importance are considered in the erasable pattern mining framework, and then a weight factor-based strong pattern pruning technique and the overestimated method satisfying the anti-monotone property are used to prevent unintended pattern losses caused by the weight factor. Yun et al. [18] proposed the first HAUPM algorithm named SHAU that considers the time factors of transactions to find recent important high average utility patterns over a data stream, where the tree-based data structure called SHAU-tree was used to store average utility information of a recent stream data batch by batch, and the batch list was used to store each node. To further decrease the number of generated candidates, a new strategy, named RUG, was proposed to minimize the values of overestimated average utilities stored in the global SHAU-tree during the accumulation of stream data.

2.2. Outlier Detection

For the clustering-based outlier detection method, Huang et al. [9] proposed a novel outlier cluster detection algorithm called ROCF that does not require the top- n parameter, where a preliminary clustering algorithm was proposed to conduct outlier cluster detection based on a Mutual Neighbors Graph constructed by connecting each point to its mutual neighbors. Then, an outlier detection approach based on the idea that outlier clusters are usually much smaller than normal clusters was proposed to detect implicit outliers. The clusters were judged as outlier clusters via decision graph instead of parameter n or α by manual set. Shi and Zhang [14] proposed a novel clustering-based outlier iterative detection method to detect the outliers and then adjusted the cluster according to the relationship between the internal relations of the clusters to improve the efficiency of the detection rate. The detection result of the clustering-based method is highly dependent on the selected clustering algorithm and is time consuming.

For the distance-based outlier detection method, Kontaki et al. [10] first proposed a novel continuous distance-based outlier detection algorithm called COD that has two versions, where the radius R was fixed, and the values of k were changing. Then, a new distance-based outlier detection algorithm called

ACOD was proposed to support the handling of multiple values of k and multiple values of R to enable the concurrent execution of different monitoring strategies. Finally, a microcluster-based outlier detection algorithm called MCOD was proposed to reduce the distance calculating times, which could be easily extended to a new edition of AMCOD to support multiple queries. Angiulli and Fassetto [1] first proposed the high memory cost distance-based outlier detection algorithm, and then they [2] proposed a strictly fixed memory requirements approximation algorithm to reduce the memory usage of the former algorithm. However, distance-based methods need to calculate the distance of each itemset, which is computationally intensive.

For the KNN-based outlier detection method, Ramaswamy et al. [13] proposed a K -nearest neighbor-based new outlier detection method to overcome some defects of the distance-based methods; it ranked every point according to the distance from its nearest neighbor point to the K -nearest neighbor point, and the top n points were determined as outliers. Although the KNN-based method is much more efficient than the distance-based methods, it is not suitable for large-scale data because its computational volume is very large.

For the density-based outlier detection method, Vries et al. [16] used a projection strategy to search the KNN outliers and then proposed an outlier detection method based on the local density to detect the outliers. Tang and He [15] presented an effective density-based outlier detection approach with local kernel density estimation (KDE). A relative density-based outlier score (RDOS) was introduced to measure the local outlierness of objects, where the density distribution at the location of an object was estimated with a local KDE method based on the extended nearest neighbors of the object. Instead of using only the k -nearest neighbors, the reverse nearest neighbors and shared nearest neighbors of an object for density distribution estimation were further considered in the proposed algorithm. However, density-based methods may face the problem of dimension disasters.

For the itemset mining-based outlier detection method, the FindFPOF method [7] was first proposed to detect the implicit outliers from a static *precise* dataset, where the judging standard was the proportion of the contained frequent itemsets to the total num-

ber of mined frequent itemsets. Because the length of each transaction was not considered in the outlier detection phase, the accuracy of outlier detection was not high enough when the length of the transactions was unfixed. The maximal frequent itemset mining-based outlier detection method was proposed by Cai et al. [4] to reduce the time cost of the outlier detection phase, where the whole outlier detection process was divided into a maximal frequent itemset mining phase and a pattern matching phase. Hemalatha et al. [8] proposed a minimal infrequent itemset mining-based outlier detection algorithm, MIFPOD, to further improve the detection accuracy. In the MIFPOD algorithm, three outlier factors, transaction weighting factor (TWF), minimal infrequent deviation factor (MIPDF) and minimal infrequent pattern-based outlier factor (MIFPOF), were designed to provide the basis for outlier detection.

3. Preliminaries and Problem Definition

In this section, we first introduce some concepts related to this paper, and then the problem definition is described to illustrate the problem that needs to be solved.

3.1. Preliminaries

Let $I = \{i_p, i_q, i_r, \dots, i_n\}$ be a set of literals called itemset and $I_s = \{i_p, i_q, \dots, i_k\}$ be a k -itemset, where $I_s \subseteq I$ and $k \in [1, n]$, I_s is a subset of I and I is the superset of I_s .

Data stream $DS = [t_1, t_2, \dots, t_m]$ contains a collection of infinite transactions, and each t_j is a subset of itemset I .

The sliding window (SW) model allows processing of only the most recent transactions from the data stream, and $|SW|$ is defined as the size of the sliding window.

Unlike the *precise* data stream, each item $\{i_j\}$ of the uncertain data stream exists with an existential probability value ($p(i_j, t_j)$), and it expresses the likelihood of $\{i_j\}$ that appeared in transaction t_j , where $0 < p(i_j, t_j) \leq 1$. If itemset X is formed by some items x , then $p(X, t_j) = \prod_{x \in X} p(x, t_j)$.

Definition 1. Support (sup): the frequency of itemset X that exists in DS is defined as *support*, $sup(X) = \sum_{j=1}^{|SW|} \prod_{x \in X} p(x, t_j)$.

Definition 2. Frequent itemset (FI): the itemset X is an FI if its *support* is not less than the predefined minimum *support* threshold min_sup .

Definition 3. Infrequent itemset (iFI): the itemset X is an iFI if its *support* is less than the predefined minimum *support* threshold min_sup .

For illustrating the above definitions more intuitively, we take the next transactions that are shown in Table 1 as an example. In this example, $|SW|$ is set to 5 and min_sup is set to 0.7.

In transaction t_1 , the existential probability of itemset $\{a\}$ is 0.8, and the existential probability of itemset $\{ab\}$ is 0.48 ($0.8 \cdot 0.6$). In the current sliding window, $sup(\{ab\}) = 0.8 \cdot 0.6 + 0.6 \cdot 0 + 0.3 \cdot 0.5 + 0 \cdot 0.5 + 0.4 \cdot 0.3 = 0.75 > 0.7$; therefore, $\{ab\}$ is a frequent itemset.

Table 1

Transaction information of a data stream

id	Transaction	id	Transaction
t_1	{a:0.8, b:0.6, d:0.3, e:0.4, f:0.2}	t_5	{a:0.4, b:0.3, c:0.6, f:0.2}
t_2	{a:0.6, c:0.9, e:0.7, f:0.1}	t_6	{a:0.7, d:0.5, e:0.4, f:0.6}
t_3	{a:0.3, b:0.5, c:0.6, e:0.2}	t_7	{b:0.6, c:0.2, d:0.4, f:0.3}
t_4	{b:0.5, d:0.2, e:0.7, f:0.3}	t_8	{c:0.5, d:0.4, e:0.7}
...

The downward closure property [4] is an important theoretical basis in FI mining to make the mining process more efficient, which can save considerable time cost on the “extension” operation.

Property 1. All nonempty subsets of frequent itemsets are frequent.

Proof. Assume s is a frequent itemset, that is $sup(s) \geq min_sup$. s' is a nonempty subset of s , which means that s' is contained in the transactions that contain s , that is, $sup(s') \geq sup(s) \geq min_sup$; therefore, s' is also frequent.

Property 2. All supersets of infrequent itemsets are infrequent.

Proof. Assume s is an infrequent itemset, s' is a superset of s , and s' is also a frequent itemset, that is, s is a subset

of s . It can be known from Property 1 that s is a frequent itemset, which is contradicted by the assumption.

3.2. Problem Definition

Based on the above definitions, the problem definition of outlier detection over uncertain data stream based on frequent itemset mining can be described as follows:

Problem definition. Given a DS and the user-specified minimal outlier threshold Φ , the task is to find the transactions whose outlier factors are less than Φ . In the frequent itemset mining phase, the task is to find the itemsets whose *support* is not less than the predefined minimal *support* of min_sup .

4. Our Proposed Approach

In this section, we propose an outlier detection method called FIM-UDSOD that is based on frequent itemset mining to detect abnormal data (outliers) over uncertain data stream.

The main steps of the FIM-UDSOD method are divided into the following: (1) frequent itemset mining stage and (2) outlier detection stage. In the frequent itemset mining stage, the matrix structure is constructed to store the data information of the uncertain data stream when the data flow in the sliding window and the infrequent itemsets are deleted to reduce the potential number of extended itemsets. In the outlier detection stage, the outliers are detected based on the created deviation factors and the outlier determination method, and the transactions whose deviation degree is less than the predefined threshold Φ are regarded as outliers. Thus, in frequent itemset mining, the creation of deviation factors and outlier deviation calculating are the core parts of outlier detection over uncertain data stream.

In this section, the first subsection presents the frequent itemset mining method, and the second subsection introduces two deviation factors and the outlier detection method based on the mined frequent itemsets.

4.1. Frequent Itemset Mining Method

To deal with the stream property, we propose a sliding window-based method called FIM-UDS to mine the most recent frequent itemsets over uncertain

data stream. Because the “extension” process is the most time-consuming process, we propose the “*upper cap*” concept to reduce the potentially extended itemsets. The whole process is split into the next 5 steps, and each step is explained in the example listed in Table 1.

1 **Matrix Construction:** In the FIM-UDS method, we use a matrix structure to store the data information of an uncertain data stream. The scale of the constructed matrix is $(|SW|+1)*m$ (m is the maximal size of items, row $(|SW|+1)$ stands for the *support* of each item). Transactions are scanned, and the probability of each item is added into matrix A in turn, the probability of item $\{i_k\}$ appearing in T_d is written as $A_{d,k}$; otherwise, $A_{d,k}$ is written as 0 if item $\{i_k\}$ does not appear in T_d . The sliding window moves to the next new transaction after finishing the current frequent itemset mining process. After constructing the matrix, the *support* of each item is calculated, and these items whose *support* is less than the predefined min_sup are not considered in the next “extension” process, and these frequent 1-itemsets are added into FLL (frequent itemset library). The example of matrix construction is listed as follows: the min_sup is set to 0.7, and the size of the sliding window is 5.

The data information of transactions t_1, t_2, t_3, t_4 and t_5 are scanned and written to matrix A successively, and then the *support* of each item is calculated. The constructed matrix A is shown in Figure 1. After calculation, 1-itemset $\{d\}$ is infrequent due to $sup(\{d\})=0.5 < 0.7$, and it is discarded to reduce the “extension” process. Then, frequent 1-itemsets of $\{a\}, \{b\}, \{c\}, \{e\}$ and $\{f\}$ are saved into FLL .

Figure 1

The construction of matrix A

	a	b	c	d	e	f
t_1	0.8	0.6	0	0.3	0.4	0.2
t_2	0.6	0	0.9	0	0.7	0.1
t_3	0.3	0.5	0.6	0	0.2	0
t_4	0	0.5	0	0.2	0.7	0.3
t_5	0.4	0.3	0.6	0	0	0.2
sup	2.1	1.9	2.1	0.5	2	0.8

2 U^{cap} and sup^{cap} calculation: The “upper cap” value (U^{cap}) for each 1-itemset is calculated before the “extension” process to reduce the potential scale of extended itemsets. The specific calculation methods of U^{cap} and sup^{cap} are listed in formula (1) and formula (2):

$$U^{cap}(x_r, t_j) = \begin{cases} p(x_r, t_j) * \prod_{m=1}^{k-1} M, & |t_j| > 1, \\ p(x_1, t_j), & |t_j| = 1 \end{cases} \quad (1)$$

$$M = \max_{r \in [1, |I|]} p(x_r, t_j).$$

$$sup^{cap}(X) = \sum_{j=1}^{|SW|} (U^{cap}(t_j) | X \subseteq t_j). \quad (2)$$

In formula (1), M is the maximal probability of the item except for itself, k is the length of the itemset (k -itemset) that needs to be extended. If $sup^{cap}(X)$ is less than the predefined min_sup , the itemset X does not require an “extension” process because any superset of infrequent itemsets is impossible. The sup^{cap} value is given in Figure 2.

In this example, $U^{cap}(a, t_1) = 0.8 * 0.6 = 0.48$. From the calculation of sup , we know that 1-itemset $\{d\}$ is infrequent; then, the values in column d are all written as 0. After calculation of sup^{cap} , $sup^{cap}(f) = 0.58 < 0.7$, that is to say, any superset that contains item $\{f\}$ cannot be frequent; therefore, item $\{f\}$ does not need to be added into potentially extended itemsets.

Figure 2

The sup^{cap} value for each 1-itemset

	a	b	c	d	e	f
t_1	0.48	0.48	0	0	0.32	0.16
t_2	0.54	0	0.63	0	0.63	0.09
t_3	0.18	0.3	0.3	0	0.12	0
t_4	0	0.35	0	0	0.35	0.21
t_5	0.24	0.18	0.24	0	0	0.12
sup^{cap}	1.52	1.31	1.17	0	1.42	0.58

3 “Frequent 2-itemset” mining: The 2-itemsets are extended by these frequent 1-itemsets with the “union” process. The “support value” (denoted as sup) calculation of each 2-itemset is “taking least”

(take the least U^{cap} value of the constituent 1-itemset) process, and then these “frequent 2-itemsets” are saved into FLL .

In the example, the 2-itemsets of $\{\{ab\}, \{ac\}, \{ae\}, \{bc\}, \{be\}, \{ce\}\}$ are extended by frequent 1-itemsets of $\{a\}, \{b\}, \{c\}, \{e\}$ with the “union” process. Then the sup^{cap} of each 2-itemset is calculated as follows:

$$sup^{cap}(\{ab\}) = 0.48 + 0 + 0.18 + 0 + 0.18 = 0.84 \text{ (frequent);}$$

$$sup^{cap}(\{ac\}) = 0 + 0.54 + 0.18 + 0 + 0.24 = 0.96 \text{ (frequent);}$$

$$sup^{cap}(\{ae\}) = 0.32 + 0.54 + 0.12 + 0 + 0 = 0.98 \text{ (frequent);}$$

$$sup^{cap}(\{bc\}) = 0 + 0 + 0.3 + 0 + 0.18 = 0.48 \text{ (infrequent);}$$

$$sup^{cap}(\{be\}) = 0.32 + 0 + 0.12 + 0.35 + 0 = 0.79 \text{ (frequent);}$$

$$sup^{cap}(\{ce\}) = 0 + 0.63 + 0.12 + 0 + 0 = 0.75 \text{ (frequent).}$$

After calculation, 2-itemset $\{bc\}$ is infrequent; it is not a potential extended itemset and any superset ($\{abc\}$ and $\{bce\}$) also cannot be frequent, therefore, the supersets of itemset $\{bc\}$ no longer need to be calculated. Then, the “frequent 2-itemsets” of $\{\{ab\}, \{ac\}, \{ae\}, \{be\}, \{ce\}\}$ are saved into FLL .

4 “Repeat (2) and (3)”: Repeat steps (2) and (3) until no frequent itemsets can be extended.

In our example, the “upper cap” value of U^{cap} for each 2-itemset is calculated with formula (1), and the sup^{cap} is calculated with formula (2). The calculation result is listed in Figure 3.

In the example, $U^{cap}(a, t_1) = 0.8 * 0.6 * 0.6 = 0.288$, other U^{cap} values are also similarly calculated. Due to $sup^{cap}(\{c\}) = 0.687 < 0.7$, itemset $\{c\}$ need not be considered in the next extended process. Then, items $\{a\}, \{b\}$ and $\{e\}$ are extended into 3-itemset $\{abe\}$ and $sup^{cap}(\{abe\}) = 0.256 + 0 + 0.072 + 0 + 0 = 0.328 < 0.7$, which is not a frequent itemset. All “extension” processes are ended because no more frequent itemsets exist in this example.

Figure 3

The $p^{cap}(x)$ and $sup^{cap}(X)$ for each 2-itemset

	a	b	c	d	e	f
t_1	0.288	0.384	0	0	0.256	0
t_2	0.486	0	0.441	0	0.567	0
t_3	0.108	0.18	0.15	0	0.072	0
t_4	0	0.245	0	0	0.175	0
t_5	0.144	0.108	0.096	0	0	0
sup^{cap}	1.026	0.917	0.687	0	1.07	0

- 5 “Frequent itemset” checking: After the “extension” process, these “frequent itemsets” that exist in FLL (except for frequent 1-itemsets) need to be checked to determine if they are truly frequent, and the infrequent itemset should be removed from FLL :

$$sup(\{ab\})=0.48+0+0.15+0+0.12=0.75 \text{ (frequent);}$$

$$sup(\{ac\})=0+0.54+0.18+0+0.24=0.96 \text{ (frequent);}$$

$$sup(\{ae\})=0.32+0.42+0.06+0+0=0.8 \text{ (frequent);}$$

$$sup(\{be\})=0.24+0+0.1+0.35+0=0.69<0.7 \text{ (infrequent);}$$

$$sup(\{ce\})=0+0.63+0.12+0+0=0.75 \text{ (frequent).}$$

With the above 5 steps, all the frequent itemsets of $\{\{a\}, \{b\}, \{c\}, \{e\}, \{f\}, \{ab\}, \{ac\}, \{ae\}, \{ce\}\}$ have been mined from the target dataset, and the detailed steps are shown in Algorithm 1.

Algorithm 1: FIM-UDS

Input: Uncertain data stream, m (largest size of items), min_sup

Output: FIs

1. construct matrix
2. add probability value into matrix
3. **for** $k=1$ to m **do**
4. calculate $sup(\{i_k\})$
5. **if** $sup(\{i_k\}) \geq min_sup$ **then**
6. $FLL \leftarrow \{i_k\}$ // frequent 1-itemsets are added into FLL
7. **end if**
8. calculate $p^{cap}(i_p, t_p)$ and $sup^{cap}(i_k)$
9. **if** $sup^{cap}(i_k) \geq min_sup$ **then**
10. **for** $k=1$ to m **do**
11. extend $\{i_x\}$ and $\{i_y\}$ into $\{i_x, i_y\}$
12. calculate $sup'(\{i_x, i_y\})$
13. **if** $sup'(\{i_x, i_y\}) \geq min_sup$ **then**
14. $FLL \leftarrow \{i_x, i_y\}$ // “frequent 2-itemsets” are added into FLL
15. **end if**
16. **end for**
17. **end if**
18. go to 8
19. **end for**
20. check if “frequent itemsets” are truly frequent
21. return FIs
22. move the sliding window
23. go to 2

4.2. Outlier Detection Method

From the definition of an outlier, we know that an outlier is data with a significant difference from other data; therefore, the outlier index of each transaction is a major indicator of outliers. He et al. [7] proposed the frequent itemset mining-based outlier detection method, but the influence of the length of the transactions on outlier detection was not fully considered. In the definition of the outlier index in the literature [10], the transaction is judged as a normal transaction if the frequent itemset outlier index is larger than the predefined threshold. However, if the length of the transaction is sufficiently long to make the proportion of the frequent itemsets to the transaction relatively small, it is not fair to judge the current transaction outlier or not based on the theory proposed in the literature [10]. Therefore, in this subsection, we proposed two outlier factors based on the mined frequent itemsets to determine the possibility of the transaction being or not being an outlier, which considers the proportion of the frequent itemsets to the length of the transaction.

Definition 4. *FIOF (Frequent Itemset Outlier Factor):* Let $DS=[t_p, t_p, \dots, t_n]$ be n transactions in the sliding window, the total number of mined frequent itemsets is $|FIL|$, *FIOF* of transaction t_i is defined as:

$$FIOF(t_i) = \frac{\sum_{X \subseteq t_i, X \in FIL} sup(X)}{|FIL|}. \quad (3)$$

The interpretation of formula (3) is as follows: if a transaction t_i contains more frequent itemsets, the *FIOF* value will be much larger, and the larger *FIOF* value indicates transaction t_i is less likely to be an outlier.

Definition 5. *TOF (Transaction Outlier Factor):* Let $DS=[t_p, t_p, \dots, t_n]$ be n transactions in the sliding window, and suppose that the size of the current transaction is m . For each transaction t_i , *TOF* is defined as:

$$TOF(t_i) = \frac{FIOF(t_i)}{m}. \quad (4)$$

The interpretation of formula (4) is as follows: if a transaction contains more frequent itemsets, its *FIOF* value is correspondingly greater; therefore, the *TOF* value is also much greater, and the higher *TOF* value indicates that transaction t_i is less likely to be an outlier.

In our approach, the TOF value is used as the basic measure to judge whether a transaction is an outlier, and the main idea of the outlier detection method over an uncertain data stream based on frequent itemset mining (FIM-UDSOD) is as follows: (1) calculate the defined values of $FIOF$ and TOF , (2) compare the TOF value with the predefined minimal threshold Φ . If $TOF(t_i)$ is less than Φ , transaction t_i is determined to be an outlier. The specific process of the FIM-UDSOD method is included in Algorithm 2.

Algorithm 2: FIM-UDSOD

Input: Data stream, Φ , n (size of $|SW|$)

Output: Outliers

1. call Algorithm 1 // mine all frequent itemsets
2. **for** $i \in [1, n]$ **do**
3. **for** X in transaction t_i and FL_L **do**
4. call formula (3)
5. call formula (4)
6. **end for**
7. **end for**
8. **if** $TOF(T_i) < F$ **then**
9. t_i is judged as outlier
10. **end if**

Next, we take the data information in Table 1 as an example to clearly explain the proposed FIM-UDSOD method. The parameters are the same as above, and Φ is set to 0.055. The number of mined frequent itemsets in this example is 9. The final calculation results are shown in Table 2.

Table 2

Specific results of outlier detection

<i>Trans</i>	<i>m</i>	<i>FIs</i>	<i>FIOF</i>	<i>TOF</i>
t_1	5	$\{a\}, \{b\}, \{e\}, \{f\},$ $\{ab\}, \{ae\}$	0.311	0.0622
t_2	4	$\{a\}, \{c\}, \{e\}, \{f\}, \{ac\},$ $\{ae\}, \{ce\}$	0.432	0.1081
t_3	4	$\{a\}, \{b\}, \{c\}, \{e\}, \{ab\},$ $\{ac\}, \{ae\}, \{ce\}$	0.234	0.0586
t_4	4	$\{b\}, \{e\}, \{f\}$	0.167	0.0417
t_5	4	$\{a\}, \{b\}, \{c\}, \{f\},$ $\{a,b\}, \{a,c\}$	0.207	0.0517

After the calculation of $TOF(t_i)$, we find that the numerical value of $TOF(t_4)$ and $TOF(t_5)$ is less than the minimal threshold Φ . Then, transactions t_4 and t_5 are determined to be outliers.

5. Experimental Results and Discussions

To evaluate the proposed FIM-UDSOD method, numerous experiments are conducted on a synthetic dataset randomly generated with lengths ranging from [6,9]. Each element of the synthetic dataset is independent and randomly selected from [1,9] with a randomly generated probability from (0,1.0), and the scale is 300 to display the outlier detection results. Then, we randomly generate several outliers to substitute for the generated elements, and the outlier element is selected from [10,15] with a random probability from (0,1.0). To test the mining efficiency of the proposed FIM-UDS method, two public datasets of *T10I4D100K* and *mushroom* downloaded from the Frequent Itemset Mining Implementations Repository (<http://fimi.cs.helsinki.fi/data/>) are used in the experiment. The existential probability range in (0,1) is randomly generated for each itemset, as suggested by the literature [11] because these datasets do not provide probability values.

The FindFPOF method [7] and MIFPOD method [8] are used as the compared methods to test the detection efficiency of the FIM-UDSOD method, which is tested with different values of min_sup and different sizes of the sliding window. To further test the mining efficiency of the proposed FIM-UDS method, the U-Apriori [5] method and UF-Growth [11] method are used as the comparison methods in the experiments. All experiments are run on a machine running Windows 7 with an Intel dual-core i3-2020 2.93 GHz processor and 4GB RAM. All algorithms are implemented using Python 3.6.

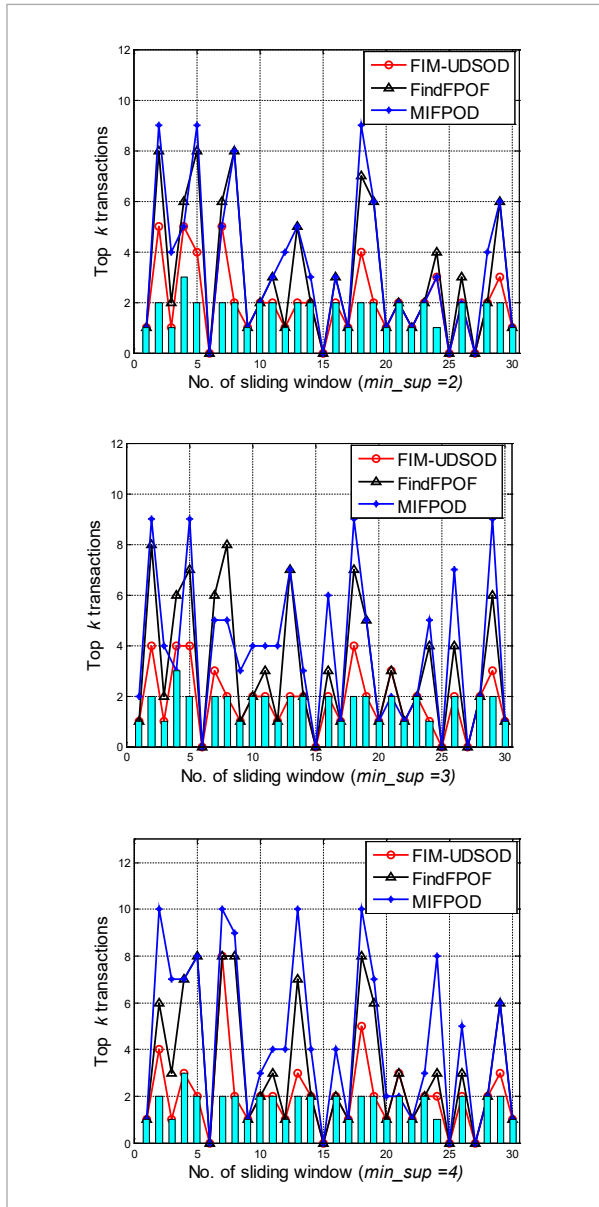
5.1. Detection Efficiency of FIM-UDSOD Method

The experiment that tests the detection efficiency of the FIM-UDSOD method is conducted with different sizes of sliding windows and different values of min_sup , where the top k transactions in this experiment indicate the selected previous k transactions with the least TOF value when all implanted outliers

are detected. The size of the sliding window in this experiment is set at 10 and 20, and the results are shown in Figures 4 and 5. The bar in the figure indicates the injected number of outliers.

Figure 4

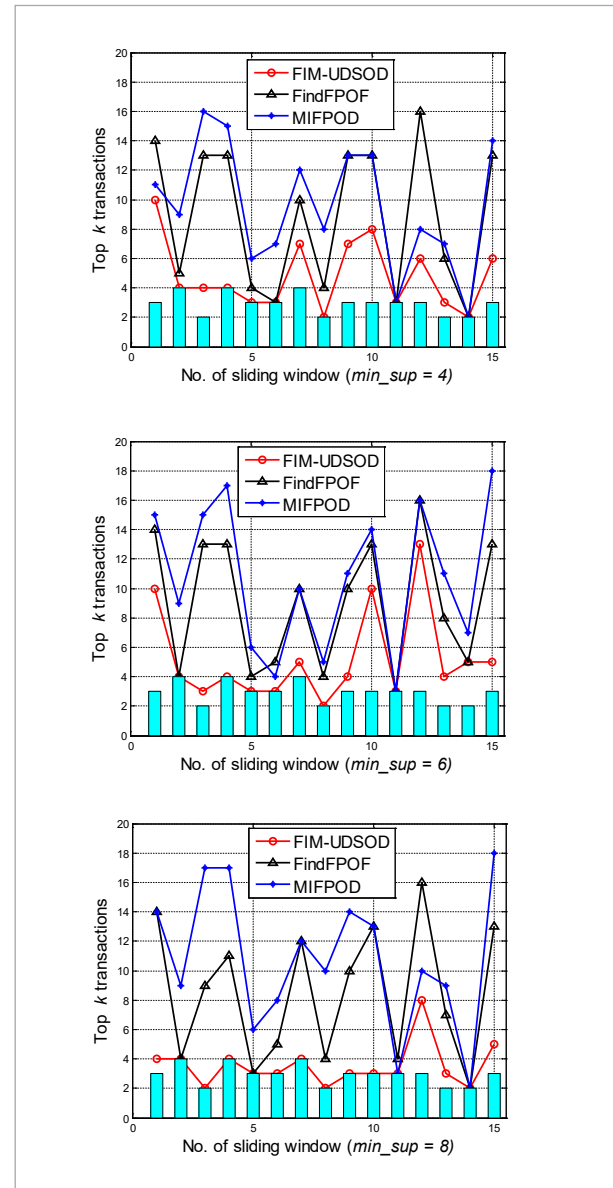
Top k transactions selected when $|SW|$ is 10



As shown in Figure 4, when the size of the sliding window is 10, the top k transactions selected by the FIM-UDSOD method are the smallest of the compared three methods. That is, the outlier detection

Figure 5

Top k transactions selected when $|SW|$ is 20



efficiency of the FIM-UDSOD method is the highest against the other two methods, and the accuracy of the MIFPOD method is relatively accurate compared to the FindFPOF method in most situations. The reason for the FIM-UDSOD method being more accurate than the FindFPOF method is that the length of transactions is taken into consideration in the outlier detection phase. Figure 5 shows the outlier detection accuracy of the compared three methods when the size of the sliding window is 20. The accuracy of the

FIM-UDSOD method is also better than the Find-FPOF method and MIFPOD method and the outlier detection of the FIM-UDSOD method is near 100% with the larger value of min_sup .

5.2. Time Cost and Memory Usage of the FIM-UDS Method

The itemset mining-based outlier detection process is divided into (1) frequent itemset mining phase and (2) outlier detection phase, where the frequent itemset mining phase requires the most time cost. This subsection shows the time cost of the FIM-UDS method with different sizes of sliding windows and different values of min_sup , where the U-Apriori [5] method and UF-Growth[11] method are compared. The experiments are conducted on two public datasets, *T10I4D100K* and *mushroom*, where *T10I4D100K* is a sparse dataset and *mushroom* is a dense dataset. The time cost of the FIM-UDS method and two compared methods on dataset *T10I4D100K* is shown in Figure 6, and the time cost on dataset *mushroom* is shown in Figure 7. The memory usage of the FIM-UDS method and the two compared methods on dataset *T10I4D100K* is shown in Figure 8, and the memory usage on the dataset *mushroom* is shown in Figure 9.

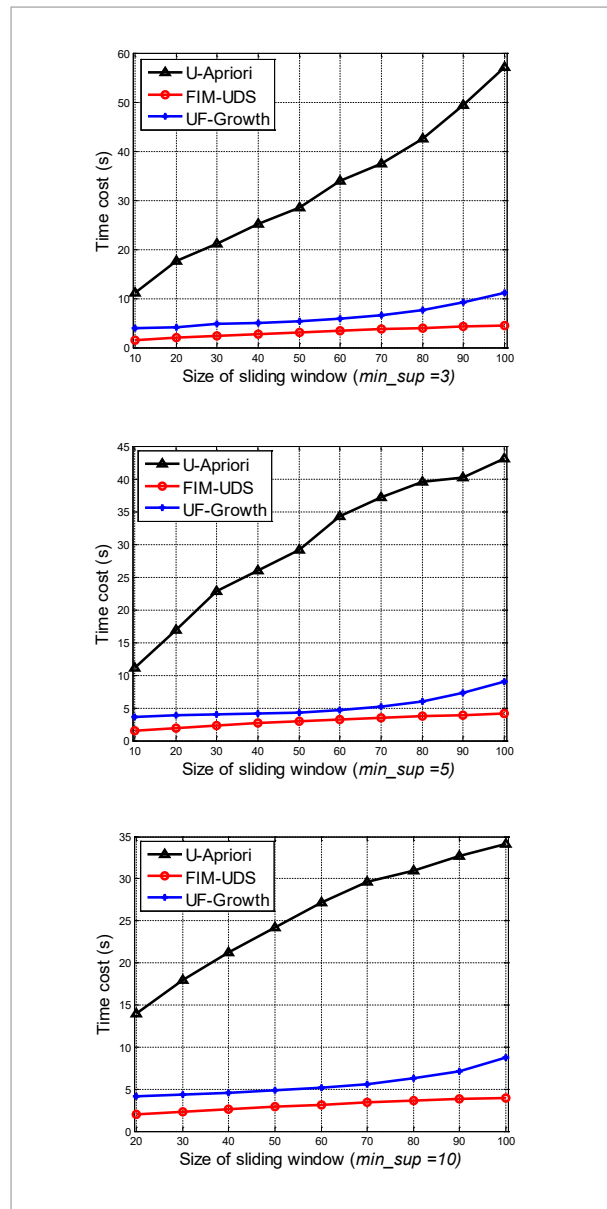
Figure 6 shows that for the sparse dataset *T10I4D100K* and a constant value of min_sup , the time cost of the FIM-UDS, U-Apriori and UF-Growth methods exhibits a growth trend with an increasing sliding window size. Compared with the other two methods, the time cost of our proposed FIM-UDS method is the smallest because a large number of itemsets are discarded in the “extension” process for the use of the “upper cap”. The time cost of the FIM-UDS method presents a stable growth trend and only increases the small ratio with an increasing sliding window size. However, the time cost of the UF-Growth method presents a small increase when the size of the sliding window is

between 10 and 70, and then it presents a sharp increase when the size of the sliding window is equal to or greater than 80. The speed of the increase in the time cost for the U-Apriori method is relatively fast.

On the sparse dataset *T10I4D100K*, when the size of the sliding window is unchanged, the time cost of the FIM-UDS method and UF-Growth method presents a very small downward trend with increasing values of min_sup . Among the three compared methods, the time cost of our proposed FIM-UDS method is the smallest with the change in the min_sup value, and the reduced

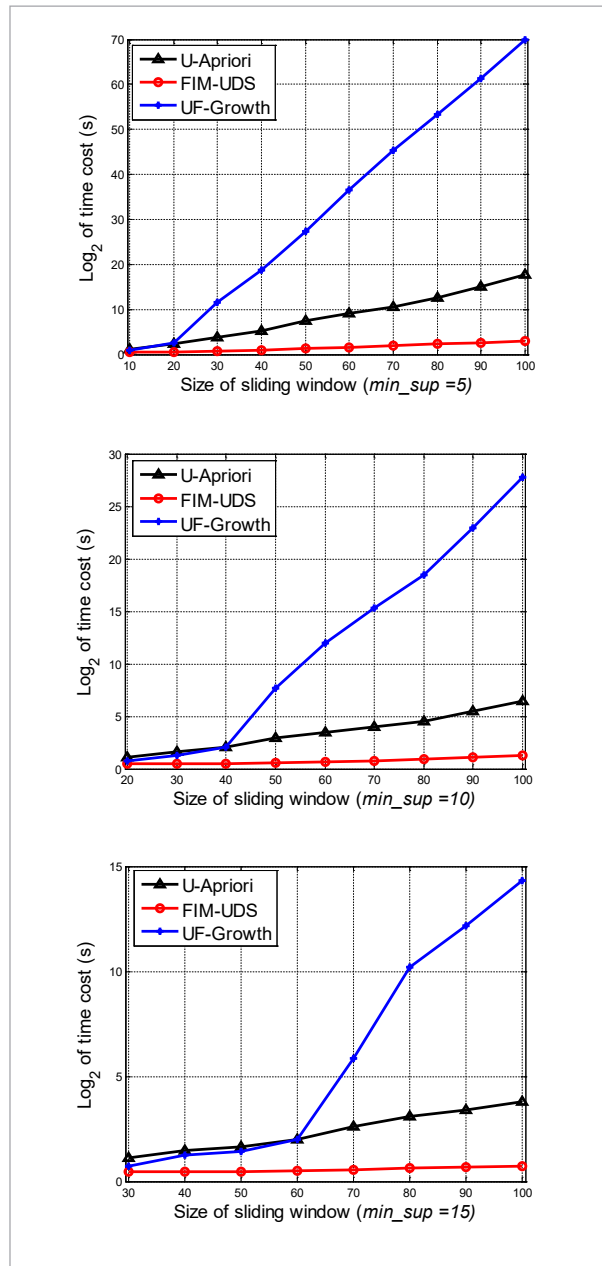
Figure 6

Time cost of three methods on sparse dataset *T10I4D100K*



range of the FIM-UDS method is also very small. As shown in Figure 7, in the dense dataset *mushroom*, when the size of the sliding window is constant, the time cost of the FIM-UDS, U-Apriori and UF-Growth methods exhibits a growth trend with increasing values of min_sup . Compared with the other two methods, the time cost of the FIM-UDS method is the smallest. Additionally, the time cost increases by only a small ratio, and the growth margin is also very small

Figure 7

Time cost of three methods on dense dataset *mushroom*

with an increasing sliding window size. However, the time cost of the UF-Growth method is relatively large, and the critical point of sudden growth increases gradually with an increase in the sliding window size. On dataset *mushroom*, when the size of the sliding window is unchanged, the time cost of the three methods presents a downward trend with the increasing values

of min_sup . Our proposed FIM-UDS method is also the most efficient method, and it is also very stable.

Figures 8 and 9 show that the peak memory usage of our proposed FIM-UDS is the lowest of the three compared methods. The peak memory usage of the FIM-UDS method shows a steady trend when the

Figure 8

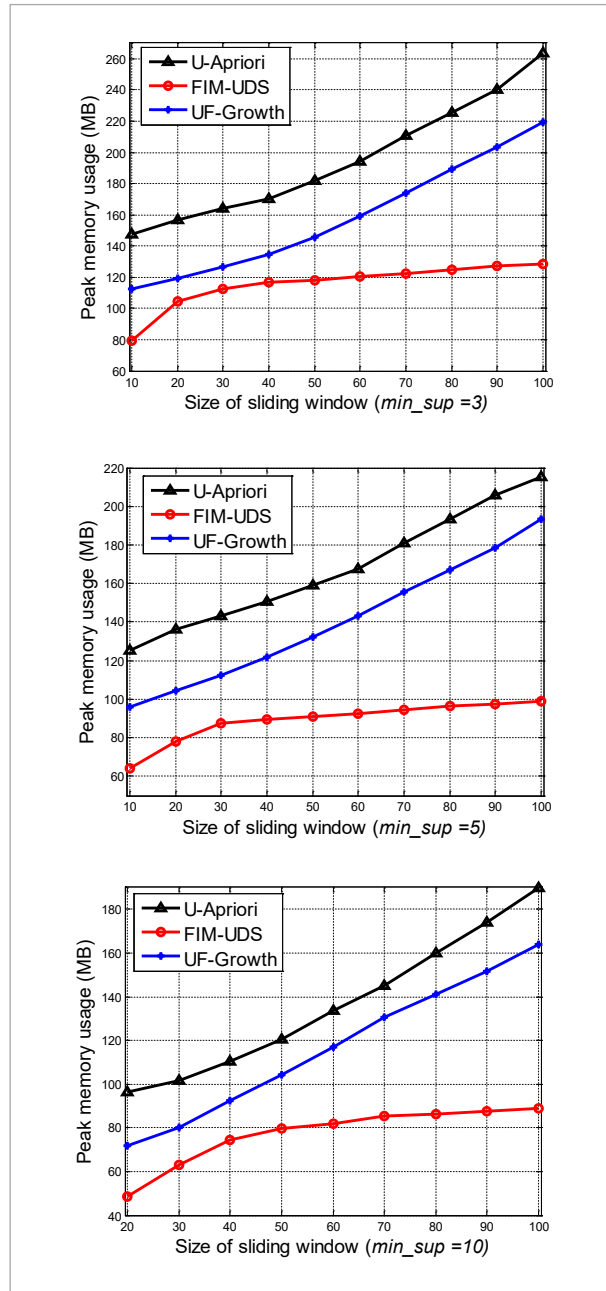
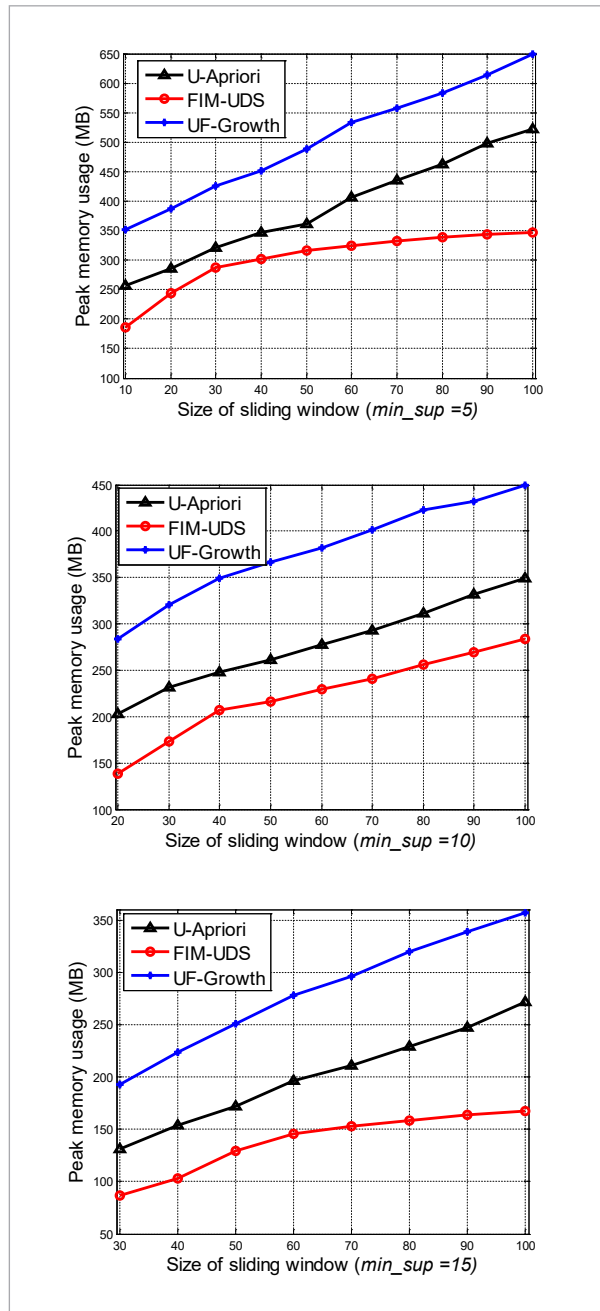
Memory usage of three methods on sparse dataset *T10I4D100K*

Figure 9

Memory usage of three methods on dense dataset *mushroom*

value of min_sup increases. More specifically, the increasing peak memory usage of the FIM-UDS method is very large when the value of min_sup is 20 and 30 on sparse dataset *T10I4D100K*, and the increasing peak memory usage of the FIM-UDS method is

very large on dense dataset *mushroom* when the value of min_sup is smaller than 40. On sparse dataset *T10I4D100K*, the peak memory usage of the U-Apriori method is the highest among the three methods, and the peak memory usage of the UF-Growth method is the highest on dense dataset *mushroom*.

6. Conclusions

With the rapid development of technology, the scale of data stream has shown an explosive growth trend in recent years, the uncertain data stream is also appearing more frequently. Unfortunately, outliers often exist accompanied with the data stream, and the existing outliers may distort the processing of the collected data stream; therefore, the issue of outliers needs to be solved. From the definition of outliers, we know that outliers rarely appear and differ from normal data; thus, outlier detection on uncertain data stream can be divided into (1) a frequent itemset mining process and (2) an outlier detection process. In this paper, we first propose an algorithm called FIM-UDS to mine the frequent itemsets over an uncertain data stream and then propose an outlier detection method called FIM-UDSOD that is based on mined frequent itemsets and design two outlier factors. More specifically, the matrix structure is used in the FIM-UDS algorithm to store the data information existing in an uncertain data stream, and the “upper cap” needs to be computed before the frequent itemset mining to reduce the number of potential extended operations. For the FIM-UDSOD method, the size of the current transaction is also considered to detect the outliers more fairly.

The experiments are conducted with different sizes of sliding windows and different values of min_sup to evaluate the performance of our proposed FIM-UDS method. The results show that the FIM-UDS method is much more effective than the U-Apriori method and the UF-Growth method. The FIM-UDSOD method is capable of detecting the existing outliers when the lengths of the transactions vary.

Acknowledgement

This work was supported by the Chinese Universities Scientific Fund under grant number 2017XD003 and the Fundamental Research Funds for the Central Universities under grant number 2018XD004.

References

1. Angiulli, F., Fassetti, F. Detecting Distance-Based Outliers in Streams of Data. Proceedings of 16th ACM Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portugal, November 6-10, 2007, 811-820. <https://doi.org/10.1145/1321440.1321552>
2. Angiulli, F., Fassetti, F. Distance-Based Outlier Queries in Data Streams: The Novel Task and Algorithms. *Data Mining and Knowledge Discovery*, 2010, 20(2), 290-324. <https://doi.org/10.1007/s10618-009-0159-9>
3. Cao, L., Yang, D., Wang, Q., Yu, Y., Wang, J., Rundensteiner, E. A. Scalable Distance-Based Outlier Detection Over High-Volume Data Streams. Proceedings of 30th IEEE International Conference on Data Engineering (ICDE 2014), Chicago, USA, March 31 - April 4, 2014, 76-87. <https://doi.org/10.1109/ICDE.2014.6816641>
4. Cai, S., Sun, R., Cheng, C., Wu, G. Exception Detection of Data Stream Based on Improved Maximal Frequent Itemsets Mining. Proceedings of 11th Springer Chinese Conference on Trusted Computing and Information Security (CTCIS 2017), Changsha, China, September 14-17, 112-125. https://doi.org/10.1007/978-981-10-7080-8_10
5. Chui, C. K., Kao, B., Hung, E. Mining Frequent Itemsets from Uncertain Data. Proceedings of 11th Springer Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007), Nanjing, China, May 22-25, 2007, 47-58. https://doi.org/10.1007/978-3-540-71701-0_8
6. Hawkins, D.M. Identification of Outliers. London: Chapman and Hall, 1980. <https://doi.org/10.1007/978-94-015-3994-4>
7. He, Z., Xu, X., Huang, Z. J., Deng, S. FP-Outlier: Frequent Pattern Based Outlier Detection. *Computer Science and Information Systems*, 2005, 2(1), 103-118. <https://doi.org/10.2298/CSIS0501103H>
8. Hemalatha, C. S., Vaidehi, V., Lakshmi, R. Minimal Infrequent Pattern Based Approach for Mining Outliers in Data Streams. *Expert Systems with Applications*, 2015, 42(4), 1998-2012. <https://doi.org/10.1016/j.eswa.2014.09.053>
9. Huang, J., Zhu, Q., Yang, L., Cheng, D, Wu, Q. A Novel Outlier Cluster Detection Algorithm Without Top-n Parameter. *Knowledge-Based Systems*, 2017, 121, 32-40. <https://doi.org/10.1016/j.knosys.2017.01.013>
10. Kontaki, M., Gounaris, A., Papadopoulos, A. N., Tsihlias, K., Manolopoulos, Y. Efficient and Flexible Algorithms for Monitoring Distance-Based Outliers Over Data Streams. *Information Systems*, 2016, 55, 37-53. <https://doi.org/10.1016/j.is.2015.07.006>
11. Leung, C. K. S., Carmichael, C. L., Hao, B. Efficient Mining of Frequent Patterns from Uncertain Data. Proceedings of 7th IEEE International Conference on Data Mining Workshop (ICDMW 2007), Omaha, USA, October 28-31, 2007, 489-494. <https://doi.org/10.1109/ICDMW.2007.84>
12. Lim, Y., Kang, U. Time-Weighted Counting for Recently Frequent Pattern Mining in Data Streams. *Knowledge and Information Systems*, 2017, 53(2), 391-422. <https://doi.org/10.1007/s10115-017-1045-1>
13. Ramaswamy, S., Rastogi, R., Shim, K. Efficient Algorithms for Mining Outliers from Large Data Sets. Proceedings of ACM SIGMOD Record (2000), Texas, USA, May 15-18, 2000, 427-438. <https://doi.org/10.1145/342009.335437>
14. Shi, Y., Zhang, L. COID: A Cluster-Outlier Iterative Detection Approach to Multi-Dimensional Data Analysis. *Knowledge and Information Systems*, 2011, 28(3), 709-733. <https://doi.org/10.1007/s10115-010-0323-y>
15. Tang, B., He, H. A Local Density-Based Approach for Outlier Detection. *Neurocomputing*, 2017, 241, 171-180. <https://doi.org/10.1016/j.neucom.2017.02.039>
16. Vries, T. D., Chawla, S., Houle, M. E. Density-Preserving Projections for Large-Scale Local Anomaly Detection. *Knowledge and Information Systems*, 2012, 32(1), 25-52. <https://doi.org/10.1007/s10115-011-0430-4>
17. Yu, J. X., Chong, Z., Lu, H., Zhang, Z., Zhou, A. A False Negative Approach to Mining Frequent Itemsets from High Speed Transactional Data Streams. *Information Sciences*, 2006, 176(14), 1986-2015. <https://doi.org/10.1016/j.ins.2005.11.003>
18. Yun, U., Kim, D., Ryang, H., Lee, G., Lee, K. M. Mining Recent High Average Utility Patterns Based on Sliding Window from Stream Data. *Journal of Intelligent & Fuzzy Systems*, 2016, 30(6), 3605-3617. <https://doi.org/10.3233/IFS-162106>
19. Yun, U., Kim, D., Yoon, E., Fujita, H. Damped Window Based High Average Utility Pattern Mining Over Data Streams. *Knowledge-Based Systems*, 2018, 144, 188-205. <https://doi.org/10.1016/j.knosys.2017.12.029>
20. Yun, U., Lee, G. Sliding Window Based Weighted Erasable Stream Pattern Mining for Stream Data Applications. *Future Generation Computer Systems*, 2016, 59, 1-20. <https://doi.org/10.1016/j.future.2015.12.012>