UDC 681.5

FP-Outlier: Frequent Pattern Based Outlier Detection

Zengyou He[1], Xiaofei Xu[1], Joshua Zhexue Huang[2], Shengchun Deng[1]

[1] Department of Computer Science and Engineering, Harbin Institute of Technology, Harbin 150001, P. R. China
zengyouhe@yahoo.com, {xiaofei, dsc}@hit.edu.cn
[2] E-Business Technology Institute, The University of Hong Kong, Pokfulam, Hong Kong, P.R.China

Abstract. An outlier in a dataset is an observation or a point that is considerably dissimilar to or inconsistent with the remainder of the data. Detection of such outliers is important for many applications and has recently attracted much attention in the data mining research community. In this paper, we present a new method to detect outliers by discovering frequent patterns (or frequent itemsets) from the data set. The outliers are defined as the data transactions that contain less frequent patterns in their itemsets. We define a measure called *FPOF (Frequent Pattern Outlier Factor)* to detect the outlier transactions and propose the *FindFPOF* algorithm to discover outliers. The experimental results have shown that our approach outperformed the existing methods on identifying interesting outliers.

Introduction

An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [25]. Mining outliers from data is an important data mining task and required in many real applications, such as credit card fraud detection, discovery of criminal activities in electronic commerce, weather prediction, marketing and customer segmentation.

In this paper, we present a new method for detecting outliers by discovering the frequent itemsets. The basic idea is simple. Since the frequent itemsets (we call them frequent patterns in this paper) discovered by the association rule algorithm [19] reflect the "*common patterns*" in the dataset, it is reasonable and intuitive to regard as outliers those data points which contain infrequent patterns. In other words, if a data object contains more frequent patterns, it means that this data object is unlikely to be an outlier because it possesses the "*common features*" of the dataset. Those infrequent patterns that are contained in few data

objects can be used as *descriptions* of outliers. We define a measure called *FPOF (Frequent Pattern Outlier Factor)* to identify the outlier objects and propose the *FindFPOF* algorithm to extract them from the data.
The main contributions of this paper can be summarized as follows:

- We define the *FPOF* measure for outlier detection. Our definition is simpler than the previous attempts to capture similar concepts [4]. This is important because the users who interpret the outliers discovered from data are usually business domain experts, not data miners [13].
- We define the *FindFPOF* algorithm to discover outliers which are determined by the total effect of frequent patterns in different sub-spaces.
- We give the *description* of the frequent patterns that outliers do not contain. These kinds of descriptions are easy to understand and provide some hints for the user for actions.
- Our approach is efficient and easy to implement because we use the existing fast frequent-pattern mining algorithms [19-21].

The rest of this paper is organized as follows. Section 2 discusses related work. In Section 3, we give 3 definitions to measure outliers based on the frequent patterns. Section 4 presents the algorithm for mining outliers. Experimental results are given in Section 5 and Section 6 concludes the paper.

Related Work and Research Motivation

Related Work

The statistics community has conducted a lot of studies on outlier mining [14]. These studies can be broadly divided into two categories. The first category is *distribution-based*, where a standard distribution is used to fit the dataset. Outliers are determined according to the probability distribution. Yamanishi et al. [10] used a Gaussian mixture model to present the normal behaviors and each datum is given a score on the basis of changes in the model. High score indicates high possibility of being an outlier. This approach has been combined with a supervised-based learning approach to obtain general patterns for outliers [12]. The second category for outlier mining in statistics is *depth-based* [15]. In the definition of depth, data objects are organized in convex hull layers in the data space according to peeling depth, and outliers are expected to be detected from data objects with shallow depth values.
Distance-based outliers are presented by Knorr and Ng [1]. A distance-based outlier in a dataset D is a data object with a given percentage of the objects in D having a distance of more than $d_{\min}$ away from it. This notion

generalizes many concepts from the distribution-based approach and enjoys better computational complexity. It is further extended based on the distance of a point from its k^{th} nearest neighbor [2]. After ranking points by the distance to its k^{th} nearest neighbor, the *top k* points are identified as outliers. Efficient algorithms for mining *top-k* outliers are given. Alternatively, in the algorithm proposed by Angiulli and Pizzuti [11], the outlier factor of each data point is computed as the sum of distances from its *k* nearest neighbors.

Deviation-based techniques identify outliers by inspecting the characteristics of objects and consider an object as an outlier if the object deviates from these features [9].

Breunig et al. [3] introduced the concept of "*local outlier*". The outlier rank of a data object is determined by taking into account the clustering structure in a bounded neighborhood of the object, which is formally defined as the "*local outlier factor*" (*LOF*). The LOCI method [13] further extended the density-based approach [3].

Clustering-based outlier detection techniques regarded *small* clusters as outliers [5] or identified outliers by removing clusters from the original dataset [8]. The authors in [16] further extended existing clustering based techniques by proposing the concept of the *cluster-based local outlier*, in which a measure for identifying the outlier-ness of each data object is defined.

Aggarwal and Yu [4] discussed a new technique for outlier detection, which finds outliers by observing the density distribution of projections from the data. That is, the data points in a local region of abnormally low density are considered to be outliers. Wei et al. [7] introduced an outlier mining method based on a hyper-graph model to detect outliers in a categorical dataset.

The replication neutral network (*RNN*) is employed to detect outliers by Harkins et al. [6]. The approach is based on the observation that the trained neutral network will reconstruct some small number of individuals poorly, and these individuals can be considered as outliers. The outlier factor for ranking data is measured according to the magnitude of the reconstruction error.

Research Motivation

As can be seen in Section 2.1, the outlier detection problem itself is not well defined and none of the existing definitions are widely accepted. Although several techniques have been proved useful in solving some outlier detection problems, the following problems still remain to be further explored and motivate our research.

Firstly, the existing techniques try to detect outliers using the distance of points in the full dimensional space. Recent research results show that in

the high dimensional space, the concept of proximity may not be qualitatively meaningful [17]. Due to the curse of dimensionality, these approaches are not appropriate for discovering outliers in a high dimensional space. Furthermore, they failed to find outliers in the subsets of dimensions. The method proposed by Aggarwal and Yu [4] considers data points in a local region of abnormally low density as outliers to conquer the curse of dimensionality. The main problem of their approach is that the outlier factor of each data object is determined *only* by the projection with the lowest density of data, without considering the effect of other projections. Moreover, their algorithm has a high computational cost. Wei et al. [7] introduced an outlier mining method based on a hyper-graph model to detect outliers from a categorical dataset. In that method, since all data points are constructed as the vertices of a hyper-graph, again it is computationally intensive.

Secondly, most studies on outlier detection are focused only on identifying outliers. In real applications, the reasons on why the identified outliers are abnormal also need to be given. Such descriptions should be intuitive and provide the user with some hints for further actions. Knorr and Ng [18] discussed the concept of intentional knowledge of distance-based outliers in terms of the subset of attributes. Their algorithms traverse all the sub-dimensions to find distance-based outliers and then the intentional knowledge. This search approach is intensive in computation and cannot provide an overall interpretability for different sub-spaces in reasoning which causes the abnormality.

Proposed Method

Agrawal formulated the problem of discovering frequent itemsets in market basket databases as follows [19]:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of m literals called *items* and the database $D = \{t_1, t_2, \ldots, t_n\}$ a set of n transactions, each consisting of a set of items from I. An itemset X is a non-empty subset of I. The length of itemset X is the number of items in X. An itemset of length k is called a *k-itemset*. A transaction $t \in D$ is said to contain itemset X if $X \subseteq t$. The *support* of itemset X is defined as the percentage of transactions in D contain X, i.e., $support(X) = ||\{t \in D \mid X \subseteq t\}|| / ||\{t \in D\}||$.

The problem of finding all *frequent itemsets* in D is defined as follows. Given a user defined threshold *minisupport*, find all itemsets with supports greater or equal to *minisupport*. Frequent itemsets are also called *frequent patterns*. The set of all frequent patterns is denoted by $FPS(D, minisupport)$, i.e., $FPS(D, minisupport) = \{X \subseteq I \mid support(X) \geq minisupport\}$.

From the viewpoint of knowledge discovery, frequent patterns reflect the "*common patterns*" that apply to many objects, or to large percentage of objects in the dataset. In contrast, outlier detection focuses on a very small percentage of data objects. Hence, the idea of making use of frequent patterns for outlier detection is very intuitive.

Definition 1: (*FPOF-Frequent Pattern Outlier Factor*) Let $D = \{t_1, t_2, \ldots, t_n\}$ be a database containing a set of n transactions with items I. Given a threshold *minisupport*, *FPS* (D, *minisupport*) is the set of all frequent patterns. For each transaction t, the *Frequent Pattern Outlier Factor of t* is defined as:

$$FPOF(t) = \frac{\sum_{X \subseteq t, X \in FPS(D, minisuppor t)} support(X)}{\| FPS(D, minisuppor t) \|} \quad (1)$$

The interpretation of formula (1) is as follows. If a transaction t contains more frequent patterns, its *FPOF* value will be big, which indicates that it is unlikely to be an outlier. In contrast, transactions with small *FPOF* values are likely to be outliers. Obviously, the *FPOF* value is between 0 and 1.

Definition 2: For each transaction t, an itemset X is said to be *contradictive* to t if $X \not\subset t$. The *contradict-ness* of X to t is defined as:

Contradict-ness (X, t) = (||X||−|| $t \cap X$||)**support* (X) (2)

In our approach, the *frequent pattern outlier factor* given in Definition 1 is used as the basic measure for *identifying* outliers. To *describe* the reasons why identified outliers are abnormal, the itemsets that are not contained in the transaction (it is said that the itemset is *contradictive* to the transaction) are good candidates for describing the reasons.

The consideration behind formula (2) is as follows. Firstly, the greater the support of the itemset X, the greater the value of *contradict-ness* of X to t, since a large support value of X suggests a big deviation. Secondly, longer itemsets give better description than that of short ones.

With definition 2, it is possible to identify the contribution of each itemset to the outlying-ness of the specified transaction. However, since it is not feasible to list all the *contradict itemsets*, it is preferable to present only the *top k contradict frequent patterns* to the end user, as given in Definition 3 below.

Definition 3: (*TKCFP-Top K Contradict Frequent Pattern*) Given D, I, *minisupport* and *FPS* (D, *minisupport*) as defined in Definition 1. For each transaction t, the itemset X is said to be a *top k contradict frequent*

pattern if there exist[1] no more than (*k-1*) itemsets whose *contradict-ness* is higher than that of X, where $X \in FPS(D, minisupport)$.

Our task is to mine *top-n* outliers with regard to the value of *frequent pattern outlier factor*. For each identified outlier, its *top k contradict frequent patterns* will also be discovered for the purpose of description.
We use the following examples to show how to compute these factors.

Example 1. Consider a ten-record customer dataset shown in Table 1. We are interested in dimensions *Age-range*, *Car*, and *Salary-level*, which are useful for analyzing the latent behavior of the customers. Assume that the minimum support is set to 0.5, we can get the set of all frequent patterns as shown in Table 2. According to formula (1), we can get the outlier factor value of each record as shown in Table 1 (the fifth column).
For each record, we also detect its *top 1 contradict frequent patterns*. Take record 1 as an example, the *contradict-ness* of {Young} to it is computed as
$(|\,|\{\text{Young}\}|\,|-|\,|\{\text{Middle, Sedan, Low}\} \cap \{\text{Young}\}|\,|)*support(\{\text{Young}\})$
$= 1*0.5=0.5$.
Similarly, the *contradict-ness* of {High} to this record is also 0.5. Hence, we list both of them.
The *top 1 contradict frequent patterns* are listed in Table 1 (the 6th column). According to the outlier definition, we can produce the *top-5* outlier candidates with respect to the *FPOF* values. Among these objects, object 5,6,8, and 10 are considered as outliers mainly because they don't satisfy the pattern {Middle, Sedan}. While for object 3 whose outlier factor value is 0.27, a little bigger than 0.17. This object doesn't satisfy the patterns {Middle, Sedan}, {Low} and {Middle}. In consideration of these patterns together, we can further explain that data records with *Car='Sedan'* usually have *Age-Range='Middle'*, but the third object is different.

Table 1. Customer Data

RID	Age-Range	Car	Salary-level	FPOF	Top *1* Contradict Frequent Patterns
1	Middle	Sedan	Low	0.35	{Young}, {High}
2	Middle	Sedan	High	0.35	{Young}, {High}
3	Young	Sedan	High	0.27	{Middle, Sedan}, {Low}, {Middle}
4	Middle	Sedan	Low	0.35	{Young}, {High}

[1] Since there could be more than one itemset having the same *contradict-ness* for a transaction, to ensure the set mined is independent of the ordering of the frequent patterns in *FPS*(*D, minisupport*), our method will mine all the itemsets whose *contradict-ness* is no less than the *k-th contradict frequent pattern*.

5	Young	Sports	High	0.17	{Middle, Sedan}
6	Young	Sports	Low	0.17	{Middle, Sedan}
7	Middle	Sedan	High	0.35	{Young}, {Low}
8	Young	Sports	Low	0.17	{Middle, Sedan}
9	Middle	Sedan	High	0.35	{Young}, {Low}
10	Young	Sports	Low	0.17	{Middle, Sedan}

Table 2. Frequent Patterns

ID	Age-Range	Support
1	{Middle}	0.5
2	{Young}	0.5
3	{Sedan}	0.6
4	{Low}	0.5
5	{High}	0.5
6	{Middle, Sedan}	0.5

Algorithm for Detecting FP-Outliers

Using the outlier factor *FPOF*, we can determine the degree of a record's deviation. In this section, we present our algorithm for detecting outliers. The algorithm *FindFPOF* for detecting outliers is listed in Fig. 1. The algorithm first gets the frequent patterns from the database using an existing association rule mining algorithm with a given *minisupport* (Step 2-3). Then, for every transaction in the database, the value of *FPOF* is computed according to Definition 1 (Step 4-11). Finally, the *top-n* FP-outliers are output with their corresponding *top-k* contradict frequent patterns (Step 12-15).

```
Algorithm FindFPOF

Input:  D                // the transaction database
        minisupport      // user defined threshold for the permissible minimal support
        top-n            // user defined threshold value for top n fp-outliers
        top-k            // user defined threshold value for top k contradict frequent patterns
Output: The values of FPOF for all transactions   //indicates the degree of deviation
        The top-n FP-outliers with their corresponding TKCFPs

01 begin
02     Mining the set of frequent patterns on database D using minisupport
03     /* the set of all frequent patterns is donated as: FPS (D, minisupport) */
04     foreach transaction t in D do begin
05         foreach frequent pattern X in FPS (D, minisupport) do begin
06             if t contains X then
07                 FPOF (t) = FPOF (t)+ support (X)
08             end if
09         end
10         return FPOF (t)
11     end
12     Output the transactions in the ascending order of their FPOF values. Stop when it
       outputs top-n transactions
13     foreach transaction t in top-n outliers do begin
14         Finds its top-k contradict frequent patterns and outputs them
15     end
16 end
```

Fig. 1. The *FindFPOF* Algorithm

The *FindFPOF* algorithm has three parts: 1) mining the frequent patterns from the database; 2) computing the value of *FPOF* for each transaction; and 3) finding the *top-n* FP-outliers with their *TKCFP* descriptions. The computational cost of the frequent-pattern mining algorithm is donated as $O(FP)$. We remark that many fast frequent-pattern mining algorithms are available [19-21] and so the computation complexity of Part 1 will be acceptable. As to Part 2, two "for loops" are required. Therefore, the computational cost of this part is $O(N*S)$, where N is number of the transactions in the database and S is the size of the set of frequent patterns. Part 3 has the computational cost of $O(N*\log N+S*(top\text{-}n)*(top\text{-}$

k)*log(*top-k*)), because finding the *top-n* outliers and the *top-k* contradict frequent patterns for each identified outlier needs a *sort* operation.
The overall computational complexity of the *FindFPOF* algorithm is:
$O(FP+N*S+N*\log N+S*(top\text{-}n)*(top\text{-}k)*\log(top\text{-}k))$ (3)
Since the *FindFPOF* algorithm can only handle datasets described in categorical attributes, to process datasets mixed with numeric data, we adopt the simple strategy of transforming the original dataset into a categorical dataset by discretizing numeric attributes. After that, *FindFPOF* algorithm is used to detect outliers in the transformed dataset. Discretization of numeric attributes will not be discussed in this paper, as there is many existing algorithms can be employed. (See the work of Liu, et al.[23] for a more recent survey about the existing discretization methods).
In our current implementation of *FindFPOF* algorithm, we employ Apriori [19] algorithm for finding the frequent itemsets. The whole algorithm is implemented using Java language with JDK 1.4 development package.

Experimental Results

A comprehensive performance study was conducted to evaluate our algorithm. In this section, we describe those experiments and their results. We ran our algorithm on real datasets obtained from the UCI Machine Learning Repository [22] to test its performance against other algorithms. At the same time, the algorithm's properties were also empirically studied.

5.1 Experiment Design and Evaluation Method

We used two real datasets to demonstrate the effectiveness of our algorithm against the *FindCBLOF* algorithm [16]. For all the experiments, the two parameters needed by the *FindCBLOF* algorithm were set to 90% and 5 separately as done in [16]. For our algorithm, the parameter *minisupport* for mining frequent patterns was fixed to 10%, and the maximal number of items in an itemset was set to 5.
As pointed out by Aggarwal and Yu [4], one way to test how well the outlier detection algorithm worked is to run the method on the dataset and test the percentage of points which belong to the rare classes. If outlier detection works well, it is expected that the rare classes would be over-represented in the set of points found. These kinds of classes are also interesting from a practical perspective.
Since we knew the true class of each object in the test dataset, we defined the objects in small classes as rare cases. The number of rare cases

identified was used as the assessment basis for comparing our algorithm with other algorithms.

Lymphography Data

The first dataset used was the Lymphography data set, which had 148 instances with 18 attributes. Among these 18 attributes, there were some numeric attributes. We disretized the numeric attributes using the automatic discretization functionality provided by the CBA [24] software. The data set contained a total of 4 classes. Classes 2 and 3 had the largest number of instances. The remained classes were regarded as rare class labels. The corresponding class distribution is illustrated in Table 3.

Table 3. Class Distribution of Lymphography Data Set

Case	Class codes	Percentage of instances
Commonly Occurring Classes	2, 3	95.9%
Rare Classes	1, 4	4.1%

Table 4 shows the results produced by the *FindFPOF* algorithm against the *FindCBLOF* algorithm. Here, the *top ratio* is ratio of the number of records specified as *top-k* outliers to that of the records in the dataset. The *coverage* is ratio of the number of detected rare classes to that of the rare classes in the dataset. For example, we let the *FindFPOF* algorithm find the *top 16* outliers with the top ratio of 11%. By examining these 16 points, we found that 6 of them belonged to the rare classes. In contrast, when we ran the *FindCBLOF* algorithm on this dataset, we found that only 4 of 16 top outliers belonged to rare classes.

Table 4: Detected Rare Classes in Lymphography Dataset

Top Ratio (Number of Records)	Number of Rare Classes Included (Coverage)	
	FindFPOF	FindCBLOF
5% (7)	5(83%)	4 (67%)
10%(15)	5(83%)	4 (67%)
11%(16)	6(100%)	4 (67%)
15%(22)	6 (100%)	4 (67%)
20%(30)	6 (100%)	6 (100%)

Furthermore, in this experiment, the *FindFPOF* algorithm performed the best for all cases and could find all the records in rare classes when the *top ratio* reached 11%. In contrast, the *FindCBLOF* algorithm achieved this goal with the *top ratio* at 20%, which was almost the twice for that of our algorithm.

Wisconsin Breast Cancer Data

The second dataset used was the Wisconsin breast cancer data set, which had 699 instances with 9 attributes. In this experiment, all attributes were considered as categorical. Each record was labeled as *benign* (458 or 65.5%) or *malignant* (241 or 34.5%). We followed the experimental method of Harkins, et al. [6] by removing some of the *malignant* records to form a very unbalanced distribution. The resultant dataset had 39 (8%) *malignant* records and 444 (92%) *benign* records[2]. The corresponding class distribution is illustrated in Table 5.

Table 5. Class Distribution of Wisconsin breast cancer data set

Case	Class codes	Percentage of instances
Commonly Occurring Classes	1	92%
Rare Classes	2	8%

With this dataset, our aim was to test the performance of our algorithm against the *FindCBLOF* algorithm and the *RNN* based outlier detection algorithm [6]. The results of the *RNN* based outlier detection algorithm on this dataset were reported in [6]. Table 6 shows the results produced by the 3 algorithms.

Table 6 Detected Malignant Records in Wisconsin Breast Cancer Dataset

Top Ratio (Number of Records)	Number of Malignant Included (Coverage)		
	FindFPOF	FindCBLOF	RNN
0% (0)	0 (0.00%)	0 (0.00%)	0 (0.00%)
1%(4)	3(7.69%)	4 (10.26%)	3 (7.69%)
2%(8)	7 (17.95%)	7 (17.95%)	6(15.38%)
4%(16)	14(35.90%)	14 (35.90%)	11(28.21%)
6%(24)	21(53.85%)	21 (53.85%)	18(46.15%)
8%(32)	28(71.79%)	27 (69.23%)	25(64.10%)
10%(40)	31(79.49%)	32 (82.05%)	30(76.92%)
12%(48)	35(89.74%)	35 (89.74%)	35(89.74%)
14%(56)	39(100.00%)	38 (97.44%)	36(92.31%)
16%(64)	39(100.00%)	39(100.00%)	36(92.31%)
18%(72)	39(100.00%)	39(100.00%)	38(97.44%)
20%(80)	39(100.00%)	39(100.00%)	38(97.44%)
25%(100)	39(100.00%)	39(100.00%)	38(97.44%)
28%(112)	39(100.00%)	39(100.00%)	39(100.00%)

[2] The resultant dataset is public available at:
http://research.cmis.csiro.au/rohanb/outliers/breast-cancer/

One important observation from Table 6 was that, compared to the *RNN* algorithm, our algorithm performed the best for all cases and never the worst. That is to say, the *FindFPOF* algorithm was more capable of effectively detecting outliers than the *RNN* algorithm. In comparison to the *FindCBLOF* algorithm, our algorithm achieved the same average performance with respect to the number of outliers identified.
Another important observation was that the *FindFPOF* algorithm found all the *malignant* records with the *top ratio* at 14%. In contrast, the *RNN* based outlier detection algorithm achieved this goal with the *top ratio* at 28%, which was twice of that of the *FindFPOF* algorithm, while the *FindCBLOF* algorithm achieved this goal with the *top ratio* at 16%.
In summary, the above experimental results on the two datasets have shown that the *FindFPOF* algorithm can discover outliers more efficiently than the other two algorithms. This demonstrates that the new concept of *FP-Outlier* is promising in practice.

5.4 Properties

In this section, we empirically study the effect of the parameter *minisupport* on the output of our algorithm. The test was aimed to find how the number of outliers changed with different values of *minisupport*. The datasets used were still the Lymphography dataset and the Wisconsin breast cancer dataset. For both datasets, the value *minisupport* varied from 10% to 90% and the top ratio was fixed to 10%.

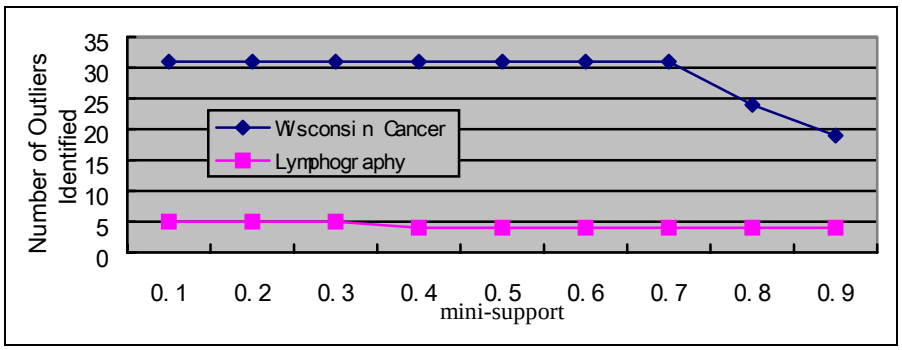

Fig.2. Number of outliers identified vs. Different value of mini-support

Fig.2 shows how the number of identified outliers changed with different values of *minisupport*. From this figure, it can be observed that, the final output of our approach is very stable, because the parameter in a wide range gives the same results. This indicates that our method is robust to the input parameter *minisupport*. Furthermore, setting a relative larger *minisupport* value to accelerate the execution of our algorithm will not affect the final results too much.

Conclusions and Future Work

Frequent pattern mining and outlier detection are two integral parts of data mining and have attracted attentions in their own fields. Based on frequent patterns, this paper has proposed a new outlier detection method. The effectiveness of the method was verified by the experimental results.

Using the same process and functionality to solve both frequent pattern mining and outlier discovery is highly desirable. Such integration will be a great benefit to business users because they do not need to worry about the selection of different data mining algorithms. Instead, they can focus on data and business solution. More importantly, some commercial data mining software do not provide the functionality of outlier discovery, hence it is easier to discover outliers directly using the frequent pattern mining results (since most commercial data mining software provide association mining module).

Several questions remain open and will be addressed in our future work:

Firstly, how to automatically assign a proper value to the parameter of *minisupport* shall be investigated.

Secondly, the number of frequent itemsets is usually huge, therefore, a number of lossless representations of frequent itemsets have recently been proposed. Two of such representations, namely the closed itemsets [26] and the generators representation [27], are of particular interest. Hence, we are planning to use these alternative representations of frequent itemsets to improve the performance of the FindFPOF algorithm.

Finally and more importantly, it is well recognized that true correlation relationships among data objects may be missed in the support-based association-mining framework. To overcome this difficulty, correlation has been adopted as an interesting measure since most people are interested in not only association-like co-occurrences but also the possible strong correlations implied by such co-occurrences. Therefore, statistical correlation based outlier detection will be another promising research direction.

Acknowledgements

The comments and suggestions from the anonymous reviewers greatly improve the paper. The High Technology Research and Development Program of China (Grant No. 2002AA413310, Grant No. 2003AA4Z2170, Grant No. 2003AA413021), the National Nature Science Foundation of China (Grant No. 40301038) and the IBM SUR Research Fund supported this research.

References

E. M. Knorr, R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In: Proc of VLDB'98, pp. 392-403, 1998.
S. Ramaswamy, R. Rastogi, S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. In: Proc of SIGMOD'00, pp. 93-104,2000.
M. M. Breunig, H. P. Kriegel, R. T. Ng, J. Sander. LOF: Identifying Density-Based Local Outliers. In: Proc. of SIGMOD'00, pp. 93-104, 2000.
C. Aggarwal, P. Yu. Outlier Detection for High Dimensional Data. In: Proc of SIGMOD'01, pp. 37-46, 2001.
M. F. Jiang, S. S. Tseng and C. M. Su. Two-phase Clustering Process for Outliers Detection. Pattern Recognition Letters, 2001, 22(6-7): 691-700.
S. Harkins, H. He, G. J. Willams, R. A. Baster. Outlier Detection Using Replicator Neural Networks. In: Proc. of DaWaK'02, pp. 170-180, 2002.
L. Wei, W. Qian, A. Zhou, W. Jin, J. X. Yu. HOT: Hypergraph-Based Outlier Test for Categorical Data. In: Proc of PAKDD'03, pp. 399-410, 2003
D. Yu, G. Sheikholeslami, A. Zhang. FindOut: Finding Out Outliers in Large Datasets. Knowledge and Information Systems, 2002, 4(4): 387-412.
A. Arning, R. Agrawal, P. Raghavan. A Linear Method for Deviation Detection in Large Databases. In: Proc of KDD'96, pp. 164-169, 1996.
K. Yamanishi, J. Takeuchi, G. Williams. On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. In: Proc of KDD'00, pp. 320-325, 2000.
F. Angiulli, C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In: Proc of PKDD'02, pp. 25-36, 2002.
K. Yamanishi, J. Takeuchi. Discovering Outlier Filtering Rules from Unlabeled Data-Combining a Supervised Learner with an Unsupervised Learner. In: Proc of KDD'01, pp. 389-394, 2001.
S. Papadimitriou, H. Kitagawa, P. B. Gibbons, C. Faloutsos. Fast Outlier Detection Using the Local Correlation Integal. In: Proc of ICDE'03, 2003.
V. Barnett, T. Lewis. Outliers in Statistical Data. John Wiley and Sons, New York, 1994.
R. Nuts, P. Rousseeuw. Computing Depth Contours of Bivariate Point Clouds. Computational Statistics and Data Analysis, 1996, 23: 153-168.
Z. He, X. Xu, S. Deng. Discovering Cluster Based Local Outliers. Pattern Recognition Letters, 2003, 24 (9-10): 1651-1660.
K. Beyer, J. Goldstein. R. Ramakrishnan, U. Shaft. When is Nearest Neighbors Meaningful? In: Proc of ICDT'99, 1999.
E. Knorr, R. Ng. Finding intentional knowledge of distance-based outliers. In: Proc of VLDB'99, pp.211-222, 1999.
R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules. In: Proc of VLDB'94, pp. 478-499, 1994.
M. J. Zaki. Scalable Algorithms for Association Mining. IEEE Trans. On Knowledge and Data Engineering, 2000, 12(3): 372~390.
J. Han, J. Pei, J. Yin. Mining Frequent Patterns without Candidate Generation. In: Proc of SIGMOD'00, pp. 1-12, 2000.
G. Merz, P. Murphy. Uci repository of machine learning databases. Technical Report, University of California, Department of Information and Computer Science: http://www.ics.uci.edu/mlearn/MLRepository.html, 1996.
H. Liu, F. Hussain, C. L. Tan, M. Dash. Discretization: An Enabling Technique. Data Mining and Knowledge Discovery, 6, pp. 393-423, 2002.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In: Proc of KDD'98, pp. 80-86, 1998.
D. Hawkins. Identification of outliers. Chapman and Hall, Reading, London, 1980.
N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In: Proc. of ICDT'99, pp. 398–416,1999.
M. Kryszkiewicz. Concise Representation of Frequent Patterns based on Disjunction–free Generators. In: Proc. of ICDM'01, pp. 305–312,2001.

Zengyou He received his M.S degree in computer science from Harbin Institute of Technology (HIT) in China in 2002. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering, HIT. His main research interests include data mining, approximate query answering, business intelligence and customer relationship management. His research work has been published in international journals such as Expert System with Applications, Information Fusion, Pattern Recognition Letters, etc.

Xiaofei Xu received both his M.S degree and Ph.D. degree in computer science from HIT in 1985 and 1988. He is currently a professor and dean of School of Computer Science and Technology, dean of National Pilot School of Software in Harbin Institute of Technology in China. He is commissioner of China Association of Science and Technology, the standing member of the council of China Computer Federation, member of Expert Group for Discipline of Computer Science and Technology in the Academic Degree Committee of the State Council of China. He is vice director of National Standard Technical Committee on Industrial Automation System and Integration, vice chairman of the Council of the China ERP Development and Technology Association. He is also senior member of Society of Manufacturing Engineer (SME) in USA, and was member of German Society of Operation Research. He has positions in the editorial committees of nine academic journals. His research fields include intelligent enterprise computing, computer integrated manufacturing systems, databases, supply chain management, e-business, knowledge engineering, etc. In recent years, he has been in charge of more than twenty Chinese national research projects and international cooperation projects, and gotten many research achievements and awards. He has published more than 200 papers in referred journals and conferences.

Joshua Zhexue Huang is the Assistant Director of E-Business Technology Institute at the University of Hong Kong. He received his Ph.D. degree from the Royal Institute of Technology in Sweden. Before joining ETI in 2000, he was a senior consultant at the Management Information Principles, Australia, consulting on data mining and business intelligence systems. From 1994 to 1998 he was a research scientist at CSIRO Australia. His research interests include data mining, machine learning, clustering algorithms, and Grid computing.

ShengChun Deng received his Ph.D. degree in computer science from HIT in 2002. He is currently an Associate Professor in the Department of Computer Science and Engineering, HIT. His main research interests include supply chain management, data mining and data warehouse.