A Fast Greedy Algorithm for Outlier Mining

Zengyou He, Xiaofei Xu, Shengchun Deng

Department of Computer Science and Engineering, Harbin Institute of Technology,

92 West Dazhi Street, P.O Box 315, P. R. China, 150001

zengyouhe@yahoo.com, {xiaofei, dsc}@hit.edu.cn

Abstract The task of outlier detection is to find small groups of data objects that are exceptional when compared with rest large amount of data. In [38], the problem of outlier detection in categorical data is defined as an optimization problem and a local-search heuristic based algorithm (LSA) is presented. However, as is the case with most iterative type algorithms, the LSA algorithm is still very time-consuming on very large datasets. In this paper, we present a very fast greedy algorithm for mining outliers under the same optimization model. Experimental results on real datasets and large synthetic datasets show that: (1) Our algorithm has comparable performance with respect to those state-of-art outlier detection algorithms on identifying true outliers and (2) Our algorithm can be an order of magnitude faster than LSA algorithm.

1. Introduction

In contrast to traditional data mining task that aims to find the general pattern applicable to the majority of data, outlier detection targets the finding of the rare data whose behavior is very exceptional when compared with rest large amount of data.Studying the extraordinary behavior of outliers helps uncovering the valuable knowledge hidden behind them and aiding the decision makers to make profit or improve the service quality. Thus, mining for outliers is an important data mining research with numerous applications, including credit card fraud detection, discovery of criminal activities in electronic commerce, weather prediction, and marketing.

A well-quoted definition of outliers is firstly given by Hawkins [1]. This definition states, "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism . With increasing awareness on outlier detection in data mining literature, more concrete meanings of outliers are defined for solving problems in specific domains [3-38].

However, conventional approaches do not handle categorical data in a satisfactory manner, and most existing techniques lack for a solid theoretical foundation or assume underlying distributions that are not well suited for exploratory data mining applications. To fulfill this void, the problem of outlier detection in categorical data is defined as an optimization problem as follows [38]: finding a subset of k objects such that the expected entropy of the resultant dataset after the removal of this subset is minimized.

In the above optimization problem, an exhaustive search through all possible solutions with k outliers for the one with the minimum objective value is costly since for n objects and k outliers there are (n,k) possible solutions. To get a feel for the quality-time tradeoffs involved, a

local-search heuristic based algorithm (LSA) is presented in [38]. However, as is the case with most iterative type algorithms, the LSA algorithm is still very time-consuming on very large datasets.

In this paper, we present a very fast greedy algorithm for mining outliers under the same optimization model. Experimental results on real datasets and large synthetic datasets show that: (1) Our algorithm has comparable performance with respect to those state-of-art outlier detection algorithms on identifying true outliers and (2) Our algorithm can be an order of magnitude faster than LSA algorithm.

The organization of this paper is as follows. First, we present related work in Section 2. Problem formulation is provided in Section 3 and the greedy algorithm is introduced in Section 4. The Empirical studies are provided in Section 5 and a section of concluding remarks follows.

2. Related Work

Previous researches on outlier detection broadly fall into the following categories.

Distribution based methods are in the first category, which are previously conducted by the statistics community [1,5,6]. They deploy some standard distribution model (e.g., normal) and flag as outliers those points that deviate from the model. Yamanishi et al. [7] used a Gaussian mixture model to present the normal behaviors and each datum is given a score based on changes in the model. This approach was combined with a supervised-based learning approach to obtain general patterns for outlier in [8].

Depth-based is the second category for outlier mining in statistics [9,10]. Based on some definition of depth, data objects are organized in convex hull layers in data space according to peeling depth, and outliers are identified as data objects with shallow depth values.

Deviation-based techniques identify outliers by inspecting the characteristics of objects and consider an object that deviates these features as an outlier [11].

Distance based method was originally proposed by Knorr and Ng [12-15]. This notion is further extended based on the distance of a point from its kth nearest neighbor [16]. Alternatively, the outlier factor of each data point is computed as the sum of distances from its k nearest neighbors in [17]. Bay and Schwabacher [18] present an algorithm with near linear time for mining distance-based outlier detection.

Density based This was proposed by Breunig et al. [19]. It relies on the local outlier factor (*LOF*) of each point, which depends on the local density of its neighborhood. Tang el at [20] introduces a connectivity-based outlier factor (*COF*) scheme that improves the effectiveness of *LOF* scheme. Three enhancement schemes over *LOF* are introduced in [21]. An effective algorithm for mining local outliers is proposed in [22]. The *LOCI* method [23] and low-density regularity method [24] further extended the density-based approach [19].

Clustering-based outlier detection techniques regarded *small* clusters as outliers [25] or identified outliers by removing clusters from the original dataset [26]. Ref. [27] proposed the concept of *cluster-based local outlier*.

Sub Space based. Aggarwal and Yu [3] discussed a new technique for outlier detection, which finds outliers by observing the density distribution of projections from the data. A frequent pattern based outlier detection method is proposed in [4], which aims at utilizing frequent patterns in different subspaces to define outliers in high dimensional space. Wei et al. [28] introduces an

outlier mining method based on a hypergraph model to detect outliers in categorical dataset.

Support vector based. Support vector novelty detector (*SVND*) was recently developed. The first *SVND* is proposed by Tax and Duin [29]. Another alternative *SVND* is proposed by Scholkopf et al. [30]. Cao et al. [31] further extended the *SVND* method. Petrovskiy [32] combine kernel methods and fuzzy clustering methods.

Neutral network based. The replicator neutral network (*RNN*) is employed to detect outliers by Harkins et al. [33,34].

In addition, the class outlier detection problem is considered in [35-37].

3. Problem Formulation

3.1 Entropy

Entropy is the measure of information and uncertainty of a random variable [2]. Formally, if X is a random variable, and $S(X)$ the set of values that X can take, and $p(x)$ the probability function of X, the entropy E(X) is defined as shown in Equation (1).

$$E(X) = -\sum_{x \in S(X)} p(x) log(p(x)) \tag{1}$$

The entropy of a multivariable vector $\hat{x} = \{X_1,...,X_m\}$ can be computed as shown in Equation (2).

$$E(\hat{x}) = -\sum_{x_1 \in S(X_1)} \cdots \sum_{x_m \in S(X_m)} p(x_1,...,x_m) log(p(x_1,...,x_m)) \tag{2}$$

3.2 Problem Formulation [38]

The problem we are trying to solve can be formulated as follows. Given a dataset D of n points $\hat{p}_1,...,\hat{p}_n$, where each point is a multidimensional vector of m categorical attributes, i.e., $\hat{p}_i = (p_i^1,...,p_i^m)$, and given a integer k, we would like to find a subset $O \subseteq D$ with size k, in such a way that we minimize the entropy of $D - O$. That is,

$$\min_{O \subseteq D} E(D - O) \tag{3}$$

subject to $|O| = k$

In this problem, we need to compute the entropy of a set of records using Equation (2). To make computation more efficient, we make a simplification in the computation of entropy of a set of records. We assume the independences of the record, transforming Equation (2) into Equation (4). That is, the joint probability of combined attribute values becomes the product of the probabilities of each attribute, and hence the entropy can be computed as the sum of entropies of the attributes.

$$E(\hat{x}) = - \sum_{x_1 \in S(X_1)} \cdots \sum_{x_m \in S(X_m)} p(x_1,...,x_m) log(p(x_1,...,x_m))$$

$$= E(X_1) + E(X_2) + ... + E(X_n) \tag{4}$$

4. The Greedy Algorithm

In this section, we present a greedy algorithm, denoted by greedyAlg1, which is effective and efficient on identifying outliers.

4.1 Overview

Our greedyAlg1 algorithm takes the number of desired outliers (supposed to be k) as input and selects points as outliers in a greedy manner. Initially, the set of outliers (denoted by OS) is specified to be empty and all points are marked as non-outlier. Then, we need k scans over the dataset to select k points as outliers. In each scan, for each point labeled as non-outlier, it is temporally removed from the dataset as outlier and the entropy objective is re-evaluated. A point that achieves *maximal entropy impact*, i.e., the maximal decrease in entropy experienced by removing this point, is selected as outlier in current scan and added to OS. The algorithm terminates when the size of OS reaches k.

4.2 Data Structure

Given a dataset D of n points $\hat{p}_1 ,..., \hat{p}_n$, where each point is a multidimensional vector of m categorical attributes, we need m corresponding hash tables as our basic data structure. Each hash table has attribute values as keys and the frequencies of attribute values as referred values. Thus, in O (1) expected time, we can determine the frequency of an attribute value in corresponding hash table.

4.3 The Algorithm

Fig.1 shows the greedyAlg1 algorithm. The collection of records is stored in a file on the disk and we read each record t in sequence.

In the initialization phase of the greedyAlg1 algorithm, each record is labeled as non-outlier and hash tables for attributes are also constructed and updated (Step 01-04).

In the greedy procedure, we need to scan the dataset for k times to find exact k outliers, i.e., one outlier is identified in each pass. In each scan over dataset, we read each record t that is labeled as non-outlier, its label is changed to outlier and the changed entropy value is computed. A record that achieves *maximal entropy impact* is selected as outlier in current scan and added to the set of outliers (Step 05-13).

In this algorithm, the key step is computing the changed value of entropy. With the use of hashing technique, in O (1) expected time, we can determine the frequency of an attribute value in

corresponding hash table. Hence, we can determine the decreased entropy value in $O(m)$ expected time since the changed value is only dependent on the attribute values of the record to be temporally removed.

```
Algorithm greedyAlg1
Input:    D    // the categorical database
          k    // the number of desired outliers
Output:   k identified outliers


/* Phase 1-initialization */
01   Begin
02       foreach record t in D
03           update hash tables using t
04           label t as a non-outlier with flag "0


/* Phase 2-Greedy Procedure */
counter = 0
05       Repeat
06           counter++
07           while not end of the database do
08               read next record t which is labeled "0     //non-outlier
09               compute the decrease on entropy value by labeling t as outlier
10               if maximal decrease on entropy is achieved by record b then
11                   update hash tables using b
12                   label t as a outlier with flag "1
13       Until counter = k
14   End
```

Fig. 1. The greedyAlg1Algorithm.

4.4   Time and Space Complexities

Worst-case analysis: The time and space complexities of the greedyAlg1 algorithm depend on the size of dataset (n), the number of attributes (m), the size of every hash table, the number of outliers (k).

To simplify the analysis, we will assume that every attribute has the same number of distinct attributes values, p. Then, in the worst case, in the initialization phase, the time complexity is $O(n*m*p)$. In the greedy procedure, since the computation of value change on entropy requires at most $O(m*p)$ and hence this phase has time complexity $O(n*k*m*p)$. Totally, the algorithm has time complexity $O(n*k*m*p)$ in worst case.

The algorithm only needs to store m hash tables and the dataset in main memory, so the space complexity of our algorithm is $O((p+n)*m)$.

Practical analysis: Categorical attributes usually have *small* domains. Typical categorical attributes domains considered for clustering consist of less than a hundred or, rarely, a thousand attribute values. An important of implication of the compactness of categorical domains is that the

parameter, p, can be regarded to be very small. And the use of hashing technique also reduces the impact of p, as discussed previously, we can determine the frequency of an attribute value in $O(1)$ expected time, So, in practice, the time complexity of greedyAlg1can be expected to be $O(n*k*m)$.

The above analysis shows that the time complexity of greedyAlg1 is linear to the size of dataset, the number of attributes and the number of outliers, which make this algorithm scalable. Previous LSA algorithm presented in [38] has the time complexity $O(n*k*m*I)$, which is much slower than our algorithm since I (the number of iterations in LSA) is usually larger than 10.

5. Experimental Results

A comprehensive performance study has been conducted to evaluate our greedyAlg1 algorithm. In this section, we describe those experiments and their results. We ran our algorithm on real-life datasets obtained from the UCI Machine Learning Repository [39] to test its performance against other algorithms on identifying true outliers. In addition, some large synthetic datasets are used to demonstrate the scalability of our algorithm.

5.1 Experiment Design and Evaluation Method

Following the experimental setup in [38], we also used two real life datasets (*lymphography* and *cancer*) to demonstrate the effectiveness of our algorithm against *FindFPOF* algorithm [4], *FindCBLOF* algorithm [27], *KNN* algorithm [16] and LSA algorithm [38]. In addition, on the *cancer* dataset, we add the results of *RNN* based outlier detection algorithm [33,34] that are reported in [33,34] for comparison, although we didn't implement the *RNN* based outlier detection algorithm.

For all the experiments, the two parameters needed by *FindCBLOF* [27] algorithm are set to 90% and 5 separately as done in [27]. For the *KNN* algorithm [16], the results were obtained using the *5-nearest-neighbour*; For *FindFPOF* algorithm [4], the parameter *mini-support* for mining frequent patterns is fixed to 10%, and the maximal number of items in an itemset is set to 5. Since the LSA algorithm and greedyAlg1 are parameter-free (besides the number of desired outliers), we don't need to set any parameters.

As pointed out by Aggarwal and Yu [3], 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 know the true class of each object in the test dataset, we define objects in small classes as rare cases. The number of rare cases identified is utilized as the assessment basis for comparing our algorithm with other algorithms.

5.2 Results on Lymphography Data

The first dataset used is the Lymphography data set, which has 148 instances with 18 attributes. The data set contains a total of 4 classes. Classes 2 and 3 have the largest number of

instances. The remained classes are regarded as rare class labels for they are small in size. The corresponding class distribution is illustrated in Table 1.

Table 1. 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 2 shows the results produced by different algorithms. 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 LSA algorithm find the *top 7* outliers with the top ratio of 5%. By examining these 7points, we found that 6 of them belonged to the rare classes.

Table 2: Detected Rare Classes in Lymphography Dataset

Top Ratio (Number of Records)	Number of Rare Classes Included (Coverage)				
	GreedyAlg1	LSA	*FindFPOF*	*FindCBLOF*	*KNN*
5% (7)	6(100%)	6(100%)	5(83%)	4 (67%)	4 (67%)
10%(15)	6(100%)	6(100%)	5(83%)	4 (67%)	6(100%)
11%(16)	6(100%)	6(100%)	6(100%)	4 (67%)	6(100%)
15%(22)	6(100%)	6(100%)	6 (100%)	4 (67%)	6(100%)
20%(30)	6(100%)	6(100%)	6 (100%)	6 (100%)	6(100%)

In this experiment, both the greedyAlg1 algorithm and LSA algorithm performed the best for all cases and can find all the records in rare classes when the *top ratio* reached 5%. In contrast, for the *KNN* algorithm, it achieved this goal with the *top ratio* at 10%, which is almost the twice for that of our algorithm.

From the above results, we can see that greedyAlg1 algorithm achieves at least the same level performance as that of LSA algorithm onLymphography data set.

5.3 Results on Wisconsin Breast Cancer Data

The second dataset used is the Wisconsin breast cancer data set, which has 699 instances with 9 attributes, in this experiment, all attributes are considered as categorical. Each record is labeled as *benign* (458 or 65.5%) or *malignant* (241 or 34.5%). We follow the experimental technique of Harkins, et al. [33,34] 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[1]. The corresponding class distribution is illustrated in Table 3.

Table 3. Class Distribution of Wisconsin breast cancer data set

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

---

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

For this dataset, we also consider the *RNN* based outlier detection algorithm [33]. The results of *RNN* based outlier detection algorithm on this dataset are reproduced from [33].

Table 4 shows the results produced by the different algorithms. Clearly, among all of these algorithms, *RNN* performed the worst in most cases. In comparison to other algorithms, greedyAlg1 preformed very well in average. Hence, this experiment also demonstrates the effectiveness of greedyAlg1 algorithm.

Although the performance of greedyAlg1 algorithm on identifying true outliers on this dataset is not so good as that of the LSA algorithm in two cases,their performance in these two are very close. And as we will show in next Section, our algorithm is very fast for larger dataset, which is more important in data mining applications.

Table 4: Detected Malignant Records in Wisconsin Breast Cancer Dataset

Top Ratio (Number of Records)	Number of Rare Classes Included (Coverage)					
	GreedyAlg1	LSA	*FindFPOF*	*FindCBLOF*	*RNN*	*KNN*
1%(4)	**4 (10.26%)**	**4 (10.26%)**	3(7.69%)	**4 (10.26%)**	3 (7.69%)	**4 (10.26%)**
2%(8)	7 (17.95%)	**8 (20.52%)**	7 (17.95%)	7 (17.95%)	6 (15.38%)	**8 (20.52%)**
4%(16)	15(38.46%)	15(38.46%)	14 (35.90%)	14 (35.90%)	11 (28.21%)	**16(41%)**
6%(24)	**22(56.41%)**	**22(56.41%)**	21 (53.85%)	21 (53.85%)	18 (46.15%)	20(51.28%)
8%(32)	27 (69.23%)	**29(74.36%)**	28(71.79%)	27 (69.23%)	25 (64.10%)	27(69.23%)
10%(40)	**33(84.62%)**	**33(84.62%)**	31(79.49%)	32 (82.05%)	30 (76.92%)	32(82.05%)
12%(48)	36(92.31%)	**38 (97.44%)**	35 (89.74%)	35 (89.74%)	35 (89.74%)	37(94.87%)
14%(56)	**39 (100%)**	**39 (100%)**	**39 (100%)**	38 (97.44%)	36 (92.31%)	**39 (100%)**
16%(64)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	36 (92.31%)	39 (100%)
18%(72)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	38 (97.44%)	39 (100%)
20%(80)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	38 (97.44%)	39 (100%)
25%(100)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	38 (97.44%)	39 (100%)
28%(112)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	39 (100%)	39 (100%)

5.4 Scalability Tests

The purpose of this experiment was to test the scalability of the greedyAlg1 algorithm against LSA algorithm when handling very large datasets. A synthesized categorical dataset created with the software[2] developed by Dana Cristofor [40] is used. The data size (i.e., number of rows), the number of attributes and the number of classes are the major parameters in the synthesized categorical data generation, which were set to be 100,000, 10 and 10 separately. Moreover, we set the random generator seed to 5. We will refer to this synthesized dataset with name of DS1.

We tested two types of scalability of the greedyAlg1 algorithm and LSA algorithm on large dataset. The first one is the scalability against the number of objects for a given number of outliers and the second is the scalability against the number of objects for a given number of objects. Both algorithms were implemented in Java. All experiments were conducted on a Pentium4-2.4G machine with 512 M of RAM and running Windows 2000. Fig. 2 shows the results of using

---

[2] The source codes are public available at: http://www.cs.umb.edu/~dana/GAClust/index.html

greedyAlg1 and LSA to find 30 outliers with different number of objects. Fig. 3 shows the results of using two algorithms to find different number of outliers on DS1 dataset.

One important observation from these figures was that the run time of greedyAlg1 algorithm tends to increase linearly as both the number of records and the number of outliers are increased, which verified our claim in Section 4.4.

In addition, greedyAlg1 algorithm is always faster than LSA algorithm. Moreover, greedyAlg1 can be at least an order of magnitude faster than LSA in most cases.

Hence, we are confident to claim that greedyAlg1 algorithm is suitable for mining very large dataset, which is very important in real data mining applications.
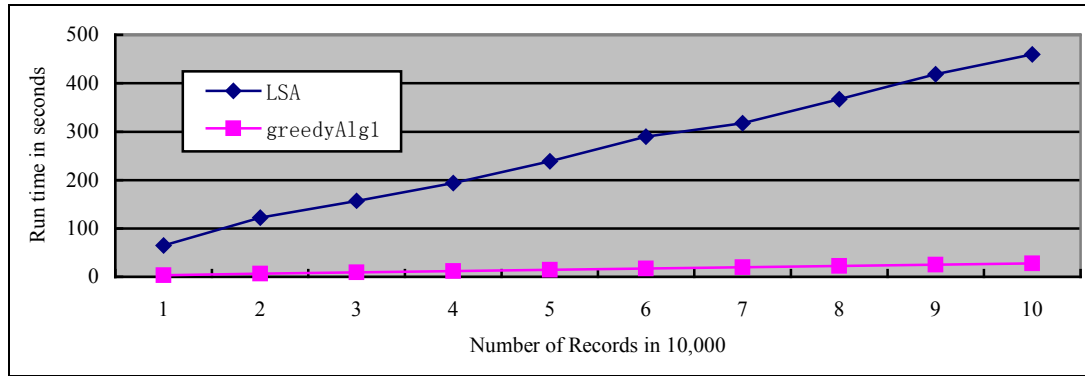

Fig. 2. Scalability of two algorithms to the number of objects when mining 30 outliers from DS1 dataset
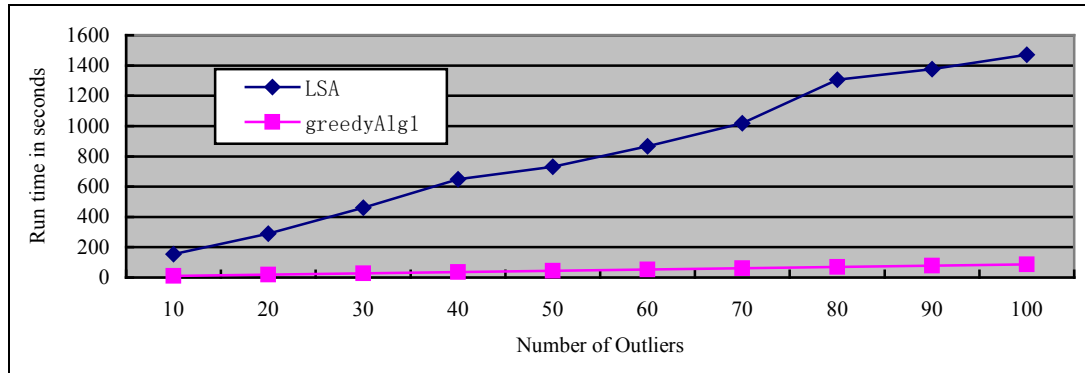

Fig. 3. Scalability of two algorithms to the number of outliers when mining outliers from 100,000 records of the DS1 dataset

6. Conclusions

Conventional outlier mining algorithms do not handle categorical data in a satisfactory manner. To fulfill this void, this paper presents a very fast greedy algorithm for mining outliers. Experimental results on real datasets and large synthetic datasets demonstrate the superiority of our new algorithm.

In future work, we will study how to automatically determine an optimal number of outliers without human intervention. Furthermore, more efficient outlier mining algorithms under the optimization model will be further addressed.

Acknowledgements

References

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

[2] C.E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 1948, pp.379-423.

[3] C. Aggarwal, P. Yu. Outlier Detection for High Dimensional Data. In: Proc of SIGMOD'01, pp. 37-46, 2001

[4] Z. He, X. Xu, J. Z. Huang, S. Deng. A Frequent Pattern Discovery Based Method for Outlier Detection. In: Proc. of WAIM'04, pp. 726-732, 2004.

[5] V. Barnett, T. Lewis. Outliers in Statistical Data. John Wiley and Sons, New York, 1994.

[6] P. Rousseeuw, A. Leroy. Robust Regression and Outlier Detection. John Wiley and Sons, 1987.

[7] 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.

[8] 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.

[9] R. Nuts, P. Rousseeuw. Computing Depth Contours of Bivariate Point Clouds. Computational Statistics and Data Analysis. 1996, vol. 23, pp. 153-168.

[10] T. Johnson, I. Kwok, R. Ng. Fast computation of 2-dimensional depth contours. In: Proc of KDD'98, pp.224-228, 1998.

[11] A. Arning, R. Agrawal, P. Raghavan. A Linear Method for Deviation Detection in Large Databases. In:Proc of KDD '96, pp. 164-169, 1996.

[12] E. Knorr, R. Ng. A unified notion of outliers: Properties and computation. In: Proc of KDD'97, pp. 219-222, 1997.

[13] E. Knorr, R. Ng. Algorithms for mining distance-based outliers in large datasets. In: Proc of VLDB'98, pp. 392-403, 1998.

[14] E. Knorr, R. Ng. Finding intentional knowledge of distance-based outliers. In: Proc of VLDB'99, pp.211-222, 1999.

[15] E. Knorr, R. Ng, T. Tucakov. Distance-based outliers: Algorithms and applications. VLDB Journal , 2000, 8(3-4): 237-253.

[16] S. Ramaswamy, R. Rastogi, S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. In: Proc. of SIGMOD'00, pp. 93-104,2000.

[17] F. Angiulli, C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In: Proc of PKDD'02, pp. 25-36, 2002.

[18] S. D. Bay, M. Schwabacher. Mining Distance Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. In: Proc of KDD'03, 2003.

[19] 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.

[20] J. Tang, Z. Chen, A. W. Fu, D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low

Density Patterns. In : Proc of PAKDD'02, pp. 535-548, 2002.

[21] A. L. Chiu, A. W. Fu. Enhancements on Local Outlier Detection. In: Proc of IDEAS'03, 2003.

[22] W. Jin, A. K. H. Tung, J. Han. Mining top-n local outliers in large databases. In: Proc of KDD'01, pp. 293-298, 2001.

[23] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, C. Faloutsos: "Fast Outlier Detection Using the Local Correlation Integal . In: Proc of ICDE'03, 2003.

[24] T. Hu , S. Y. Sung. Detecting pattern-based outliers. Pattern Recognition Letters, 2003, 24 (16): 3059-3068.

[25] M. F. Jiang, S. S. Tseng , C. M. Su. Two-phase Clustering Process for Outliers Detection. Pattern Recognition Letters, 2001, 22(6-7): 691-700.

[26] D. Yu, G. Sheikholeslami, A. Zhang. FindOut: Finding Out Outliers in Large Datasets. Knowledge and Information Systems, 2002, 4(4): 387-412.

[27] Z. He, X. Xu, S. Deng. Discovering Cluster Based Local Outliers. Pattern Recognition Letters, 2003, 24 (9-10): 1641-1650

[28] L. Wei, W. Qian, A. Zhou1, W. Jin, J. X. Yu. HOT: Hypergraph-Based Outlier Test for Categorical Data. In: Proc of PAKDD'03, pp. 399-410, 2003.

[29] D.M.J Tax, R.P.W. Duin. Support vector data description. Pattern Recognition Letters, 1999,20 (11-13), 1191-1199.

[30] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, R.C. Williamson. Estimating the support of a high dimensional distribution. Neural Computation, 2001, 13 (7): 1443-1472.

[31] L. J. Cao, H. P. Lee, W. K. Chong. Modified support vector novelty detector using training data with outliers. Pattern Recognition Letters, 2003, 24 (14): 2479-2487.

[32] M. Petrovskiy: A Hybrid Method for Patterns Mining and Outliers Detection in the Web Usage Log. In: Proc of AWIC'03, pp.318-328, 2003.

[33] 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.

[34] G. J. Willams, R. A. Baster, H. He, S. Harkins, L. Gu. A Comparative Study of RNN for Outlier Detection in Data Mining. In: Proc of ICDM'02, pp. 709-712, 2002.

[35] Z. He, S. Deng, X. Xu. Outlier detection integrating semantic knowledge. In: Proc. of WAIM'02, pp.126-131, 2002.

[36] S. Papadimitriou, C. Faloutsos. Cross-outlier detection. In: Proc of SSTD'03, pp.199-213, 2003.

[37] Z. He, X. Xu, J. Z. Huang, S. Deng. Mining class outlier: concepts, algorithms and applications in CRM. Expert System with Applications, 2004, 27(4): 681-697.

[38] Z. He, X. Xu, S. Deng. An optimization model for outlier detection in categorical data. In: Proc. of 2005 International Conference on Intelligent Computing (ICIC'05), Lecture Notes in Computer Science 3644, pp.400-409, 2005.

[39] G. Merz, P. Murphy. Uci repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html, 1996.

[40] D. Cristofor, D. Simovici. Finding median partitions using information-theoretical-based genetic algorithms. Journal of Universal Computer Science, 2002, 8(2): 153-172.