社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007 − DBS − 143 (71)
2007／7／3

Detecting Outliers in High Dimensional Datasets with Examples

Yuan LI† and Hiroyuki KITAGAWA†,††

† Graduate School of Systems and Information Engineering
†† Center for Computational Sciences
University of Tsukuba
Tennoudai 1-1-1, Tsukuba, Ibaraki, 305-8573 Japan

Abstract Many real applications of outlier detection can be utilized in various fields. Most of such applications process high dimensional datasets. In one of our previous work, we proposed an Example-Based outlier detection method making use of outlier examples to detect outliers that have similar " outlier-ness " to these examples in high dimensional datasets. However, it employs meshes for deciding outliers, and all points within the same cell are regarded as normal objects or outliers. Thus, sometimes normal objects may be detected as outliers, and vice versa. Distance-Based outlier (DB-Outlier) detection is a simple and commonly used approach, besides it can detect real outliers by calculating the distance between data points. In this paper, we introduce a method to detect DB-Outliers in high dimensional datasets based on outlier examples, and show some comparison experiments between our new method and the old one.

Key words Outlier, High-dimensional Data, Example

1. Introduction

Detecting abnormal data points (outliers) from large datasets is useful in many applications like credit card fraud detection, network robustness analysis and so on. Traditional outlier detection methods generally define algorithms to detect outliers based on distance or density such as Clustering-Based method, Distance-Based method [3] and Density-Based method [7]. In our previous work [10], we have already explained the merits of Distance-Based method (DB-Outlier detection). Therefore we try to propose a method to detect DB-Outliers in high dimensional datasets.

Outlier detection problems are well solved for low dimensional datasets. However, most existing methods fail to deal with high dimensional spaces including Distance-Based method. It has been shown that in high dimensional space data is sparse, so every data point becomes a good outlier candidate. Thus Distance-Based or Density-Based methods lose their significance when the dimensionality of a dataset is high [6]. It has been certified that Subspace-Based method [5] [9] is constructive in solving the curse of the dimensionality which is unavoidable. In other words, studying the performance of data in low dimensional subspaces is promissing in detecting outliers from high dimensional datasets.

On the other hand, most traditional techniques need to predefine some parameters which often contain hidden view of outliers. Whereas it is difficult for users to decide such relevant parameters in advance. Users are familiar with their problem domain and they probably know what kind of objects can be recognized as outliers. Therefore, it is easier for users to produce some outlier examples that include hidden user view of outliers than to predefine relevant parameters. Outlier examples can release users from the hard work of deciding parameters beforehand. Example-Based method is presented to be useful in discovering the hidden user view of outliers in [1].

An Example-Based outlier detection method (one of our previous work) introduced in [2] is performed to be reasonable in detecting outliers from high dimensional datasets. This method makes use of Sparisity-Based outlier (SB-Outlier) detection [5] to find outliers in high dimensional datasets. Sparisity-Based method utilizes meshes for deciding outliers. All data points within the same cubes are marked as normal objects or outliers. Hence, there is a crucial problem that with the effect of meshes, sometimes normal objects may be detected as outliers, and vice versa. That is to say SB-Outlier detection can not pick out real outliers based on users' viewpoint precisely. Furthermore, the the method proposed in [2] still needs an extra parameter to be decided in advance except outlier examples. The extra parameter affects the results of the detection seriously, and it is also difficult to be predefined.

In our previous work [10] we presented a method to detect DB-Outliers in high dimensional datasets with outlier exam-

ples and did some experiments to show our proposed method works well on both synthetic and real datasets. In this paper we plan to give a brief introduction on our proposed method (see the detailed explain in [10]) and mainly present some comparison experiments between DB-Outlier detection and SB-Outlier detection. From the results of these comparison experiments we can discover the merits and demerits of our proposed method clearly. Although the processing time of DB-Outlier detection is longer than SB-Outlier detection, DB-Outlier detection can produce better results; moreover the input of our proposed method is only outlier examples.

2. Overview of SB-Outlier

In this section we give an overview of SB-Outlier detection. The notion of SB-Outlier is defined by Aggarwal and Yu [5]. They detect outliers by examining projections whose density are abnormally low. "outlier-ness" is measured by the sparsity coefficient namely outliers are detected based on sparsity coefficient. For the sake of defining such low density projections, they divide each attribute of the data into ϕ equi-depth ranges. In each range, there are fraction $h = \frac{1}{\phi}$ data. In order to differentiate the alphabet from the one used in our proposed method [10], we define fraction by h instead of f (in [2] [5] they use f to denote the fraction). If the dimensionality of a dataset is k, there are h^k cubes after dividing and each cube has a fraction h^k data of a dataset. Let N stand for data size, then the expected number of objects with in a k-D (k dimensional) cube should be $N * h^k$. Figure 1 is an example of a 2-D space to explain the dividing.

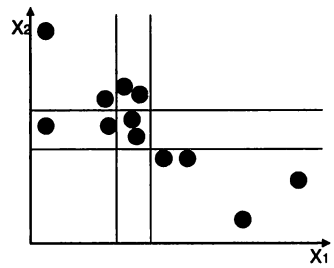


Figure 1 Equi-depth Ranges

In this 2-D space, we divide x_1 and x_2 into 3 parts separately. Each part of x_1 or x_2 has 4 data points. So the fraction $h = \frac{1}{\phi} = \frac{1}{3}$ and the expected number of data points within a cube should be $N * h^2 = 12 * (1/3)^2 = 1.333$

If the attributes are independent statistically, the fraction of data points in a k-D cube is h^k. However, in real applications, attributes are usually differ from statistically independent, so the actual fraction of data points is different greatly from the average behavior [2]. Let $n(C)$ denote the number of data points in a k-D cube C. If the attributes are independent statistically, then the number of data points in a k-D cube is a Bernoulli random variable with probability h^k of presence because of the equi-depth grids. Consequently, the number of data points in a cube approximately abide by the binomial distrtibution [2]. Since the average number of objects within a k-D cube is $N * h^k$ and the standard deviation is $\sqrt{N * h^k * (1 - h^k)}$. The sparsity coefficient $S(C)$ is defined as follows [5]:

$$S(C) = \frac{n(C) - N * h^k}{\sqrt{N * h^k * (1 - h^k)}} \quad (1)$$

Cubes which contain data points less than average number ($N * h^k$) have negative sparsity coefficients. From the formulation we can easily conclude that the less the data points within a k-D cube, the smaller the sparisity coefficient is. Besides, lower dimensionality of the cube leads to smaller sparisity coefficient [2]. Consequently, SB-Outlier detection detect objects as outliers in some special cubes whose sparisity coefficient is sparser than or as sparse as those cubes containing outlier examples.

The sparisity coefficient is affinitive with parameter ϕ. If parameter ϕ is improper for the dataset's distribution, the sparisity coefficient is not suitable for detecting outliers. For instance, there are 1000 data in a dataset. If ϕ=1000, then every object can be detected as an outlier. Users usually can not confirm the value of ϕ, so they need test some possible values to find a proper one. If the data size is large, it will take a large quantity of time to make sure the suitable value of ϕ. Moreover, SB-Outlier detection recognizes every data point in a cube as normal data or outliers. However, sometimes outliers and normal data may stay in the same cube. In this case, SB-Outlier detection is unable to separate outliers from normal data. This disadvantage often causes bad results. In our experiments we compare the quality of results between SB-Outlier and DB-Outlier detections.

3. Proposed Method

Here we just give a brief introduction on our proposed method. The sepcific description is presented in [10]. At first we plan to show the notion and two detection algorithms of DB-Outlier.

3.1 Distance-Based Outlier

The definition of DB-Outlier studied here is the same as Knorr and Ng's work [3]:

An object O in a dataset T is a DB(p, D)-Outlier if at least fraction p of the objects in T lie greater than distance D from O.

The parameter p is the minimum fraction of objects in a dataset that must be outside an outlier's D-neighborhood. For the convenience of explanation and calculation, we employ another parameter M which denotes the maximum portion of data points within the D-neighborhood of an outlier. M can be calculated by the function below:

$$M = N(1-p) \qquad N : datasize \tag{2}$$

It is conspicuous that the concept of DB-Outlier is reasonable defined for any dimensional spaces. The DB-Outlier detection can be elucidated as detecting those objects that have less than M neighbors within their D-neighborhood.

There are two algorithms for mining DB-Outliers in large datasets. One is the simple algorithm which operates based on the notion of DB-Outlier, and the other is the Cell-Based algorithm [3].

- **Simple Algorithm**

The simple algorithm executes as examining the nearest neighborhood centered at each object. As soon as M data points are found in an object' D-neighborhood, the searching engine will mark this object as a non-outlier, and then begin to examine another object. It takes much time to confirm whether an object is a real DB-Outlier. In the next paragraph we will give a reference to another algorithm that has a better processing time than the simple algorithm and can also detect real DB-Outliers.

- **Cell-Based Algorithm**

The Cell-Based algorithm (see [3] for detailed direction) makes use of cell structure's properties for ruling out non-outliers quickly. It quantizes all the data objects into a space that has been partitioned into cells or squares. These cells or squares have a particular size which correlates with parameter D, therefore majority of non-outliers can be eliminated from outlier candidates by counting the number of data points in these cells. However, the Cell-Based algorithm gives worse performance than the simple one if the dimensionality of a dataset is high. It has been proved that the Cell-Based algorithm is no better than the simple algorithm when the dimensionality of the examined dataset is more than 4 [3]. The reason is that Cell-Based algorithm will take much more quantizing time if the dimensionality is high.

3.2 Procedure of Proposed Method

Our object is to propose a approach without complicated inputs to detect outliers based on users'viewpoint. The input of our method is only the outlier examples. For the sake of detecting outliers that have simialr "outlier-ness" characteristics to outlier examples in high dimensional datasets, we first look for the most suitable subspace where outlier examples are isolated more significantly than in any other subspaces. Users actually want to pick out exceptions that have some (not so many) abnormal attributes inasmuch as the dimensionality of the most suitable subspace is ususlly low. After discovering such special subspace, we search objects as sparse as outlier examples based on Distance-Based algorithms in the most suitable subspace. If the dimensionality of the best subspace is no more than 4 we prefer to use the Cell-Based algorithm, otherwise use the simple algorithm. We employ a GA (Genetic Algorithm) to search the most suitable subspace in a shortter time.

(1) **Detecting Most Suitable Subspace with a GA**

At first, we detect a low dimensional subspace where outlier examples stand more lonely than in any other subspaces with a Genetic Algorithm.

(2) **Parameter Selection**

After discovering the most suitable subapace, some pairs of parameters p and D are selected automatically to detect objects which have similar "outlier-ness" characteristic to outlier examples in this subspace.

(3) **Outlier Report**

Due to different parameters, we may detect some different outliers in the most suitable subspace with the dissimilar automatically selected parameters (p, D). Therefore, we do not only report outliers detected in the most suitable subspace, but also report the " outlier-ness " degree of each outlier.

Since the detailed explains have been introduced in [10], in this paper we want to show some further experiments to compare the results of our proposed method and the one proposed in [2].

4. Experiments and Results

In this section we give two groups of experiments to show our proposed method is effective and efficient in detecting outliers from high dimensional datasets. In the first group of experiments, we test our proposed method on both synthetic and real datasets. It is the same as the experiments in [10]. We list the results of the first group of experiments here to present the feasibility of our proposed method. In the second group of experiments, we compare our proposed method with the approach introduced in paper [2]. All of our experiments were run on a Microsoft Windows XP machine having 1GB of main memory. The details of our experiments is shown below.

4.1 First Group of Experiments

Table 1 is the description of synthetic and real datasets used in examining our proposed method. Dim denotes the dimensionality of a dataset. The real datasets is the abalone data obtained from the UCI machine learning repository [4].

First we randomly select 3 objects as normal data (g=3)

Table 1 Synthetic and Real Datasets

Dataset	Dim	Description
Synthetic Data	16	20,012 data which are normally distributed in 16 dimensions.
Abalone Data	8	Abalone data, obtained from the UCI machine learning repository, 4177 examinations of abalones with 8 attributes.

and do the same calculation 5 times (q=5) to compute the average A_{NO}. After discovering the most suitable subspace with the help of a GA, 10 pairs of parameters (p, D) are selected to detect DB-Outliers. At last report both outliers found in the most suitable subspace and their "outlier-ness" degrees. Due to the small data size of abalone dataset the risk of randomly choosing an outlier as a normal data may be a little higher. We also do the same experiment with (g=8) for abalone dataset and compare the results with (g=3).

Figure 2 is the distribution of data points in two subspaces of synthetic dataset. The triangles denote outlier examples. It is clearly that outlier examples isolated more significantly in subspace (a) than in subspace (b). Actually subspace (a) is the best subspace. Our proposed method detected the most suitable subspace (a) correctly and report DB-Outliers and the "outlier-ness" degrees in subspace (a). Figure 3 shows the ourliers of synthetic dataset detected in the most suitable subspace whose "outlier-ness" degrees are over 70%. If users want to find outliers that are closer to the major data, they can choose outliers with lower "outlier-ness" degrees such as 60% or 50%.

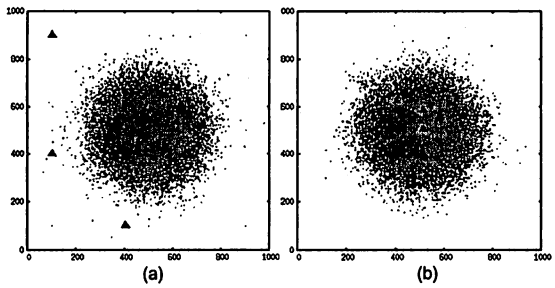


Figure 2 Outlier Examples of Synthetic Dataset

Figure 4 is the distribution of abalone data points with outlier examples in Diameter-Whole Weight subspace and Length-Height subspace. For the real dataset, we input outlier examples all isolated in the Diameter-Whole Weight subspace. That means the Diameter-Whole Weight subspace is the most suitable subspace. Our proposed method detected the correct subspace of abalone dataset successfully when g=8 and in most of trials when g=3. Figure 5 shows the DB-Outliers whose "outlier-ness" degrees are over 70% in

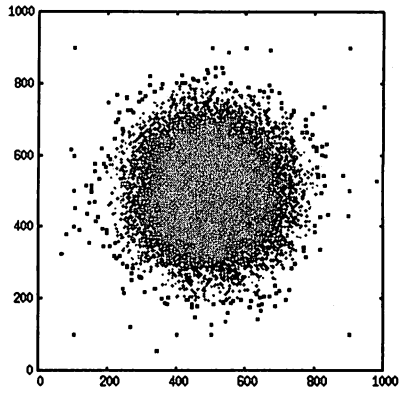


Figure 3 Detected Outliers

the most suitable subspace of the abalone dataset.

Table 2 Summary of the Proposed Method.

Dataset	p_Size	Trials	Acc	Convergence	g
Synthetic	120	10	100%	80%	3
Abalone	50	10	70%	90%	3
Abalone	50	10	90%	90%	8

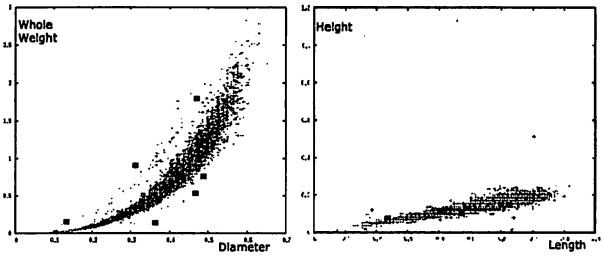


Figure 4 Outlier Examples of Real Dataset

The first group of experiments are used to test the effectiveness of our proposed method. We did the same experiments 10 times and the summary of the results is listed in Table 2. p_Size denotes the GA's population size, and Acc is the accuracy of discovering the most suitable subspace within 10 trials. Convergence is the abbreviation of convergence criterion for each dataset and g is the number of objects that we randomly select as normal data points to calculate A_{NO}.

4.2 Second Group of Experiments

The second group of experiments compare the results between our proposed method and the SB-Outlier detection proposed in paper [2] in the terms of time and quality. Since both of the two methods were certified to be effective with

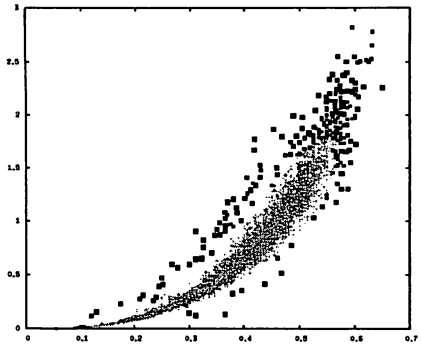


Figure 5 Detected Outliers

real datasets we only give the results of SB-Outlier detection shown in Figure 7(c). We just want to compare the processing time and quality of the two method clearly, we show the comparative results operated with a synthetic dataset. Table 3 is the description of two synthetic datasets used in the second group of experiments. Dim still denotes the dimensionality of a dataset. The real datasets is the abalone data obtained from the UCI machine learning repository [4].

Table 3 Datasets of Second Group of Experiments

Dataset	Dim	Description
Dataset I	16	10,000 data which are normally distributed in 16 dimensions.
Dataset II	16	10,000 data which are uniformly distributed in 14 dimensions and holding a group of outliers in the remainder oblique line distributed 2-D subspaces.

For Sparisity-Based mehtod, it needs another input as a predefined parameter ϕ (see [2] for comprehensive overview) to detect SB-Outliers, and the parameter have relevant effect on results. Figure 6 shows the outliers detected by our proposed method. Figure 7(a) (b) is the results of the method proposed in paper [2].

From the outcomes of the two methods, we can clearly see that our proposed method has better quality when treating with datasets whose distributions are not oblique to coordinate axes, and the unpredictable parameter ϕ plays an important role in detecting SB-Outliers. However, the processing time of our method is not so good as the one proposed in paper [2] despite we make use of Cell-Based algorithm to detect DB-Outliers when the dimensionality of the most suitable subspace is no more than 4. Due to the different results of the two methods with synthetic dataset I, we only give the summary of the comparison experiments with synthetic

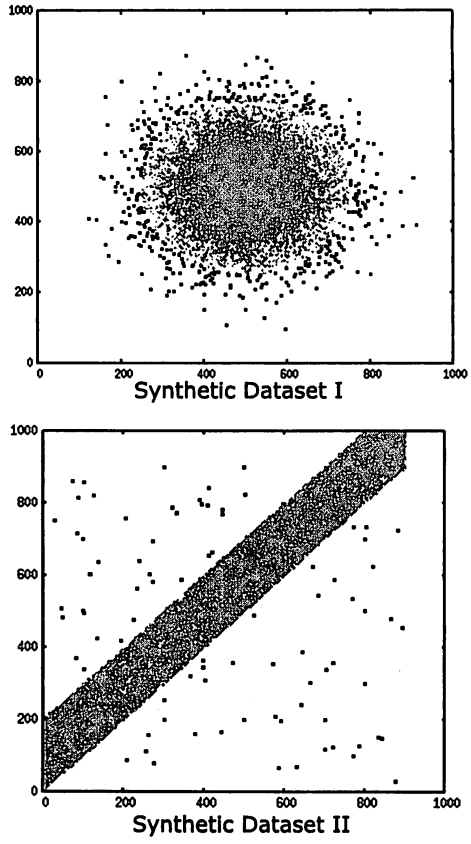


Figure 6 DB-Outlier

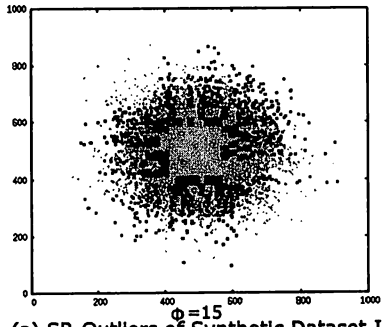
dataset I in Table 4. We also do the same experiments 10 times to compute average processing time. Subspace_Time means the average time of 10 trials for discovering the most-suitable subspace and Outlier_Time is the average time for detecting outliers. Accuracy denotes the accuracy of the two methods of discovering the most suitabel subspace within 10 times. We try to find the most suitable subspace with a GA which means the dimensionality of solutions can not always smaller than 4. Therefore we only use simple algorithm to discover the most suitable subspace.

5. Conclusions and Future Work

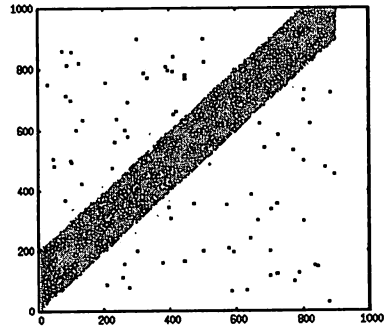
In this paper we discussed two methods of detecting outliers from high dimensional datasets. Most traditional techniques need users to decide some parameters beforehand. In fact these parameters are generally important factors for detecting outliers and not easy to be predefined such as (p, D) in our proposed method. Users are often expert in their problem domain, so it is not difficult for them to make some outlier examples. In most cases users even have a few outlier examples in hand and want to find more objects that have

Table 4 Comparison Results.

Type of Outlier	p_Size	Subspace_Time (ms)	Outlier_Time (ms)	Convergence	Accuracy
DB-Outlier (Simple Algorithm)	80	95586	317925	90%	100%
DB-Outlier (Cell-Based Algorithm)	80	-	147156	90%	-
SB-Outlier (ϕ=15)	80	11881	79	90%	60%
SB-Outlier (ϕ=22)	80	18625	125	90%	50%



(a) SB-Outliers of Synthetic Dataset I



(b) SB-Outliers of Synthetic Dataset II

Figure 7 SB-Outlier

similar "outlier-ness" characteristics to these outlier examples. Consequently, making good use of these outlier examples is an outstanding idea to avoid predefining some significant parameters. However the Example-Based method introduced in paper [2] still need a key parameter ϕ to be decided in advance. Becides, this parameter does not only effect the quality of results but also has a great influence on processing time. It is easy to conclude from Table 4 that the bigger the ϕ is the longer the processing time is.

We have tested our proposed method on both synthetic and real datasets in [10]. That time we considered all the outlier examples as real outliers. In reality, dummy outliers should be allowed appearing in outlier examples. In the future we plan to do research on improving the accuracy of discovering the most suitable subspace and processing time, and provide new idea on tolerating bad outlier examples.

Acknowledgements

This research has been supported in part by the Grant-in-Aid for Scientific Research from JSPS(#18200005) and MEXT(#19024006).

References

[1] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier By Example. *Proc. PAKDD 2004, LNAI 3056*, pp.222-234, 2004.

[2] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-Based Outlier Detection for High Dimensional Datasets. *IPSJ Transactions on Databases*, Vol. 46, No. SIG5, pp.120-129, 2005.

[3] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. *Proc. VLDB*, pp.392-403, 1988.

[4] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[5] C. C. Aggarwal and P. S. Yu. Outlier Detection for High Dimensional Data. *Proc. SIGMOD Conf.*, pp.37-46, 2001.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is Nearest Neighbors Meaningful? *Proc. Int. Conf. Database Theory*, pp.217-235, 1999.

[7] M. M. Breuning, H. P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. *Proc. SIGMOD Conf.*, pp.93-104, 2000.

[8] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. *Addison Wesley*, 1989.

[9] C. C. Aggarwal and P. S. Yu. An Effective and Efficient Algorithm for High-dimensional Outlier Detection. *The VLDB Journal*, Vol. 14, No. 2, pp.211-221, 2005.

[10] Y. Li and H. Kitagawa. DB-Outlier Detection by Example in High Dimensional Datasets. *Proc. Proc. 3rd IEEE International Workshop on Databases for Next-Generation Researchers (SWOD)*, 2007.