Outlier Detection Adaptive to Users' Intentions

Cui ZHU[†], Hiroyuki KITAGAWA[††], Spiros PAPADIMITRIOU[†††], and Christos FALOUTSOS[†††]

† Graduate School of Systems and Information Engineering, University of Tsukuba
†† Institute of Information Sciences and Electronics, University of Tsukuba
††† School of Computer Science, Carnegie Mellon University

Abstract Outlier detection has many applications like fraud detection, medical analysis, etc. Recently, several methods for finding outliers in large datasets have been reported. These existing techniques traditionally detect based on some prescribed definitions of outliers. However, it is very difficult for a user to decide the definition of outliers in prior. Usually, they have a few outlier examples in hand, and want to find more objects just like those examples. To solve this problem, we propose a novel method to detect outliers adaptive to users' intensions implied by the outlier examples. This is, to the best of our knowledge, the first that detect outliers based on user-provided examples. Our experiments on both synthetic and real datasets show that the method has the ability to discover outliers that match the users' intentions.

Key words data mining, knowledge discovering, personalization, profile, user interface

1. Introduction

Outlier detection has many applications like fraud detection, medical analysis, etc. Methods for finding outliers in large datasets are drawing increasing attention.

Intuitively, an object is an ”outlier” or ”abnormal” if it is in some way ”significantly different” from its ”neighbors”. Different answers to what constitutes a ”neighborhood”, how to determine ”difference” and whether it is ”significant” would lead to various sets of objects defined as outliers.

There have been various interpretations of the notion of the outlier (e.g., distance-based [9], density-based [3], etc.) in different scientific communities. Consequently several approaches have been proposed. As we can see, not everyone has the same idea of what constitutes an outlier.

For easy understanding, let us see a concrete example shown in Figure 1. In this data set, there are a large sparse cluster, a small crowded one and some obvioulsy isolated objects. When we look from a wide scale, only the isolated objects (circle dots) should be regarded as outliers because their neighborhood densities are very low compared with objects in eithor the large or the small cluster. However, when we consider the neighborhood of a middle scale, objects fringing with the large cluster (diamond dots) can also be regarded as outliers. Furthermore, objects fringing with the small cluster (cross dots) become outliers when we focus on neighborhood of a small scale. This illustrate that different sets of objects should be regarded as outliers if we consider from different scale of neighborhood.

In most circumstances, users are experts in their problem domain and not in outlier detection. It is very difficult for a user to decide the definition of outliers in prior. Usually, they have a few outlier examples in hand, which may “describe” their intentions and want to find more objects that exhibit “outlier-ness” characteristics just like those examples.

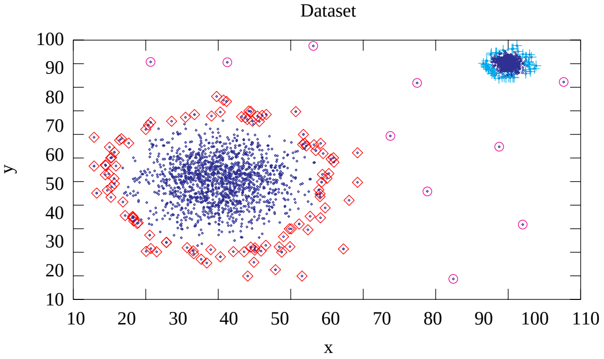

Figure 1 Illustration of different kinds of outliers in a dataset.

However, to the best of our knowledge, none of the existing methods can directly incorporate user examples in the discovery process. We present here a novel method that detects outliers adaptive to users' intensions implied by the outlier examples.

The remainder of the paper is organized as follows: In section 2, we discuss related work on outlier detection. In section 3, we discuss the measurement of “outlier-ness” and the different properties of outliers. Section 4 presents the proposed method in detail. Section 5 reports the experimental evaluation on both synthetic and real dataset. Finally, Section 6 concludes the paper.

2. Related Work

In essence, outlier detection techniques traditionally employ unsupervised learning processes. The several existing approaches can be broadly classified into the following categories: *(1) Distribution-*

based approach, [10], [14]. *(2) Depth-based approach.* [13]. *(3) Clustering approach.* [1]. *(4) Distance-based approach.* [3], [4], [12]. All of the above approaches regard being an outlier as a binary property. They do not take into account both the degree of "outlier-ness" and where the "outlier-ness" is presented. *(5) Density-based approach,* [9]. They introduced a local outlier factor (LOF) for each object, indicating its degree of "outlier-ness." When the value of the parameter MinPts is changed, LOF can be estimated in different scopes. *(6) LOCI.* We proposed the multi-granularity deviation factor (MDEF) and LOCI in [11]. MDEF measures the "outlier-ness" of objects in neighborhoods of different scales. LOCI examines the MDEF values of objects in all ranges. Even though the definition of LOF and MDEF can capture "outlier-ness" in different scales, these difference of scales were not taken into account.

Another outlier detection method was developed in [8], which focuses on the discovery of rules that characterize outliers, for the purposes of filtering new points later.This is a largely orthogonal problem. Outlier scores from SmartSifter are used to create labeled data, which are then used to find the outlier filtering rules.

In summary, all the existing methods are designed to detect outliers based on some prescribed criteria for outliers. This is the first proposal for outlier detection using user-provided examples.

3. Outlier-ness

In order to understand the users' intentions and the "outlier-ness" they are interested in, a first, necessary step is measuring the "outlier-ness." We employ the multi-granularity deviation factor (MDEF) [11] for this purpose, which is capable of measuring "outlier-ness" of objects in the neighborhoods of different scales (i.e., radii).

Here we describe some basic terms and notation. Let the r-neighborhood of an object p_i be the set of objects within distance r of p_i. Let $n(p_i, \alpha r)$ and $n(p_i, r)$ be the numbers of objects in the αr-neighborhood ($counting\ neighborhood$) and r-neighborhood ($sampling\ neighborhood$) of p_i respectively.(1) Let $\hat{n}(p_i, r, \alpha)$ be the average, over all objects p in the r-neighborhood of p_i, of $n(p, \alpha, r)$.

Definition (MDEF). For any p_i, r and α, the $multi - granularity\ deviation\ factor$ $(MDEF)$ at radius (or scale) r is defined as follows:

$$MDEF(p_i, r, \alpha) = \frac{\hat{n}(p_i, r, \alpha) - n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \quad (1)$$

Intuitively, the MDEF at radius r for a point p_i is the relative deviation of its local neighborhood density from the average local neighborhood density in its r-neighborhood. Thus, an object whose neighborhood density matches the average local neighborhood density will have an MDEF of 0. In contrast, outliers will have MDEFs far from 0.

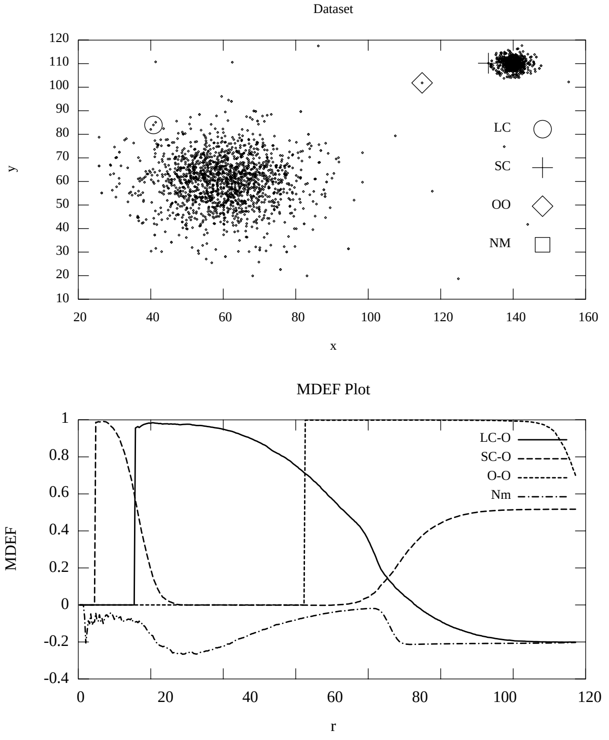

Figure 2 Illustrative dataset and MDEF plots.

In our paper, the MDEF values are examined (or, sampled) at a wide range of sampling radii r, $r_{min} \le r \le r_{max}$, where r_{max} is the maximum distance of all object pairs in the given dataset and r_{min} is determined based on the number of objects in the r-neighborhood of p_i. In our experiments, for each p_i in the dataset, r_{min} for p_i (denoted by $r_{min,i}$) is the distance to its 20-th nearest neighbor. This is a reasonable choice which effectively avoids introduction of statistical errors in MDEF estimates in practice.

To better illustrate MDEF, we give some examples. Figure 2 shows a dataset which has mainly two groups: a large, sparse cluster and a small, dense one, both following a Gaussian distribution. There are also a few isolated points. We show MDEF plots for four objects in the dataset.

- Consider the point in the middle of the large cluster, NM (box dot), (at about $x = 60$, $y = 57$). The MDEF value is low at *all* scales, indecating that the object can be always regarded as a normal object in the dataset.
- In contrast, for the other three objects, there exist situations where the MDEFs are very large, some times even approaching 1. This shows that they differ significantly from their neighbors in *some* scales.

Even though all three objects in Figure 2 can be regarded as outliers, they are still different, in that they exhibit "outlier-ness" at different scales.

- The outlier in the small cluster, SC (cross dot), (at about $x = 133$, $y = 110$), exhibits strong "outlier-ness" in the scale about $r = 5$.
- On the other hand, the outlier of the large cluster, LC (circle

(1): In all experiments, $\alpha = 0.5$ as in [11].

```
Input:
    Set of outlier examples: $E$
    Fraction of outliers: $F$
    Dataset: $D$

Output:
    Outliers like examples

Algorithm:
    // Feature extraction step:
    For each $p_i \in D$
        For each $j$ ($0 \leq j \leq n$)
            Compute MDEF value $m_{ij}$
    // Classification step:
    $POS$ := $E$
    $NEG$ := strongest negatives
    $P$ := $D$
    Do {
        $P'$ := $P$
        SVM := construct_SVM ($POS, NEG$)
        ($P$, $N$) := SVM.classify ($D$)
        $NEG$ := $N$
    } while ($|P| \geq F * |D|$ and $|P| \neq |P'|$)
    return $P'$
```

Figure 3 The overall procedure of the proposed method

dot), (at about $x = 40$, $y = 84$), exhibits strong "outlier-ness" in the range from $r = 10$ to $r = 30$.

- For the isolated outlier, OO (diamond dot), (at about $x = 115$, $y = 102$), its MDEF value stays at 0 up to almost $r = 22$, indicating that it is an isolated object. Then, it immediately displays a high degree of "outlier-ness."

4. Proposed Method

The proposed method detects outliers based on user-provided examples and a user-specified fraction of objects to be detected as outliers in the dataset. The method performs in two stages: feature extraction step and classification step.

4.1 Feature Extraction Step

The purpose of this step is to map all objects into the MDEF-based feature space, where the MDEF plots of objects capturing the degree of "outlier-ness," as well as the scales at which the "outlier-ness" appears, are represented by vectors. Let D be the set of objects in the feature space. In this space, each object is represented by a vector: $O_i = (m_{i0}, m_{i1}, \ldots, m_{in}),\ O_i \in D$, where $m_{ij} = MDEF(p_i, r_j, \alpha r),\ 0 \leq j \leq n,\ r_0 = min_k(r_{min,k}),\ r_n = r_{max}, r_j = \frac{r_n - r_0}{n} j + r_0$.

4.2 Classification Step

After the user-provided examples, as well as the entire, unlabeled dataset are mapped into feature space, the next crucial step is to find an efficient and effective algorithm to discover the "hidden" outlier concept that the user has in mind.

We use an SVM (Support Vector Machine) classifier to learn the "outlier-ness" of interest to the user and then detect outliers which match this. Traditional classifier construction needs both positive and negative training data. However, it is too difficult and also a burden for users to provide negative data.

However, the proposed algorithm addresses this problem and can learn only from the examples and the unlabeled data (i.e., the rest of the objects in the dataset). The algorithm uses the marginal property of SVMs. In this sense, it bears some general resemblance to PEBL [5], which was also proposed for learning from positive and unlabeled data. However, in PEBL, the hyperplane for separating positive and negative data is set as close as possible to the set of given positive examples. In the context of outlier detection, the positive examples are just examples of outliers, and it is not desirable to set the hyperplane as in PEBL. The algorithm here decides the final separating hyperplane based on the fraction of outliers to be detected. Another difference from PEBL is that strong negative data are determined taking the characteristics of MDEF into consideration.

The classification step consists of the following five sub-steps.

Negative training data extraction sub-step All objects are sorted in descending order of $max_j(m_{ij})$. Then, from the objects at the bottom of the list, we select a number of (strong) negative training data equal to the number of examples. Let the set of strong negative training data be NEG. Also, let the set of examples be POS.

Training sub-step Train a SVM classifier using POS and NEG.

Testing sub-step Use the SVM to divide the dataset into the positive set P and negative set N.

Update sub-step Replace NEG with N, the negative data obtained in the testing sub-step.

Iteration sub-step Iterate from the training sub-step to the updating sub-step until the ratio of the objects in P converges to the fraction specified by the user. The objects in the final P are reported to the user as detected outliers.

Figure 3 summarizes the overall procedure of the proposed method.

5. Experiments

In this section, we describe our experimental methodology and the results on both synthetic and real data. The results illustrate the variousness for users' intensions and also demonstrate the effectiveness of our method.

5.1 Experimental procedure

Our experimental procedure is as follows:

(1) To simulate interesting outliers, we start by selecting objects which represent "outlier-ness" at some scales under some conditions, for instance, $\bigwedge_q (min_q, max_q, Cond_q, K_q)$, where $(min_q, max_q, Cond_q, K_q)$ stands for the condition that $(m_{ij}\ Cond_q\ K_q)$ for some j such that $min_q \leq j \leq max_q$, where $Cond_q$ could be either "$>$" or "$<$".

(2) Then, we randomly sample y% of the outliers to serve as examples that would be picked by a user,[2] and "hide" the remainders.

(3) Next, we detect outliers using the proposed method.

(4) Finally, we compare the detected outliers to the (known) simulated set of interesting outliers. Evaluations are based on precision/recall measurements:

$$Precision = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ predictions} \qquad (2)$$

$$Recall = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ data} \qquad (3)$$

We use the LIBSVM [7] implementation for our SVM classifier. In all experiments, we use polynomial kernels and the same SVM parameters[3]. Therefore, the whole processes can be done automatically.

5. 2 Datasets and Sets of Interesting Outliers

We do experiments on three synthetic and one real dataset to evaluate the proposed method. Table 1 shows the descriptions of all datasets.

Table 1 Description of synthetic and real datasets.

Dataset	Description
Uniform	A 6000-point group following an uniform distribution.
Ellipse	A 6000-point ellipse following a Gaussian distribution.
Mixture	A 5000-point sparse Gaussian cluster, a 2000-point dense Gaussian cluster and 10 randomly scattered outliers.
Medical	Offered by PKDD'99 Discovery Challenge [6], 7950 GPT and GLU examinations of patients in Chiba University hospital.

Table 2 shows all the sets of interesting outliers along with the corresponding discriminants used as the underlying outlier concept in our experiments. In the table, for instance, the discriminant (1, 40, $>$, 0.9) means that objects are selected as interesting outliers when their MDEF values are greater than 0.9 in the range of radii from 1 to 40. The number of interesting outliers is also shown in Table 2.

We always randomly sample 10% ($y = 10$) of the interesting outliers to serve as user-provided examples and "hide" the rest. Detected interesting outliers are those returned by the classifier.

5. 3 Experimental Results

Uniform dataset Figure 4 shows the outliers detected by our method on the uniform dataset. Although one might argue that no objects from an (infinite!) uniform distribution should be labeled as outliers, the objects at the fringe or corner of the group are clearly "exceptional" in some sense. On the top row, we show the interesting outliers, outlier examples supposed to be picked by users and the detected results for case U-Fringe. The bottom row shows those for case U-Corner (see Table 2 for a description of the cases). Note that the chosen features can capture the notion of both "edge" and "corner" and, furthermore, the proposed method can almost perfectly detect outliers adaptive to these various intensions implied by the different users' examples!

Ellipse dataset We simulate three kinds of interesting outliers for the ellipse data set: (i) the set of fringe outliers whose MDEF values are examined at a wide range of scales, (ii) those mainly spread at the long ends of the ellipse which display outlier-ness in two ranges of scales (from 15 to 25 and from30 to 40), and (iii) mainly in the short ends, which do *not* show strong outlier-ness in the scales from 35 to 40. The output of the proposed method is shown in Figure 5. Again, the features can capture several different and interesting types of outlying objects and the proposed method again discovers the underlying outlier notion!

Mixture dataset Here we also mimic three categories of interesting outliers: (i) the set of outliers scattered along the fringe of both clusters, (ii) those maily spread along the fringe of the large sparse cluster, and (iii) those mainly in the small dense cluster. The results of detection are shown in Figure 6.

Medical dataset In the real medical dataset, we mimic two kinds of intention for outliers: The first group (case M-Sector) is the set of outliers scattered along the sector part of the whole dataset. These objects display a high degree of "outlier-ness" when we examine from a wide scale. The second group of outlying objects (case M-Origin) are those who concentrate around the origin. They are discovered when we focus into small scales. The results of detection are shown in Figure 7.

For all datasets, Table 3 shows the precision and recall measurements for the proposed method, using polynomial kernels. It also shows the number of iterations needed to converge in the learning step. In Table 3, all the measurements are averages of ten trials.

In almost all cases of synthetic datasets, the proposed method detects interesting outliers with both precision and recall reaching 80–90%. In the cases of the real medical dataset, the two measurements are a little worse. But it still achieves 59% precision and 71% recall in the worst case (case M-Sector of medical dataset). And the number of iterations is always small (less than 10) in all cases.

6. Conclusion

Outlier detection is an important, but tricky problem, since the intention of outlier definition often depends on the user and/or the dataset. We propose to solve this problem by bringing the user in the loop, and allowing him or her to give us some examples that he or she considers as outliers. Experiments on both real and synthetic data demonstrate that the method can succesfully incorporate these

[2] In all experiments, $y = 10$.

[3] For the parameter C (the penalty imposed on training data that fall on the wrong side of the decision boundary), we use 1000, i.e., a high penalty to mis-classification. For the polynomial kernel, we employ a kernel function of $(u' * v + 1)^2$.

Table 2　Interesting Outliers and the Discriminants.

Dataset	Cases			
	Label	Discription	Condition	# of Interesting Outliers
Uniform Dataset	U-Fringe	Fringe	(0.3, 0.6, >, 0.4)	330
	U-Corner	Corner	(1, 2, >, 0.5)	274
Ellipse Dataset	E-Fringe	Fringe	(5, 30, >, 0.85)	214
	E-Long	Long Ends	(15, 25, >, 0.8) (30, 40, >, 0.6)	137
	E-Short	Short Ends	(5, 15, >, 0.8) (35, 40, <, 0.6)	157
Mixture Dataset	M-All	All	(1, 40, >, 0.9)	162
	M-Large	Large Cluster	(15, 40, >, 0.9)	114
	M-Small	Small Cluster	(1, 10, >, 0.9)	49
Medical Dataset	M-Sector	Sector Part	(100, 600, >, 0.97)	163
	M-Origin	Origin Part	(0, 40, >, 0.84) (140, 200, <, 0.2)	75

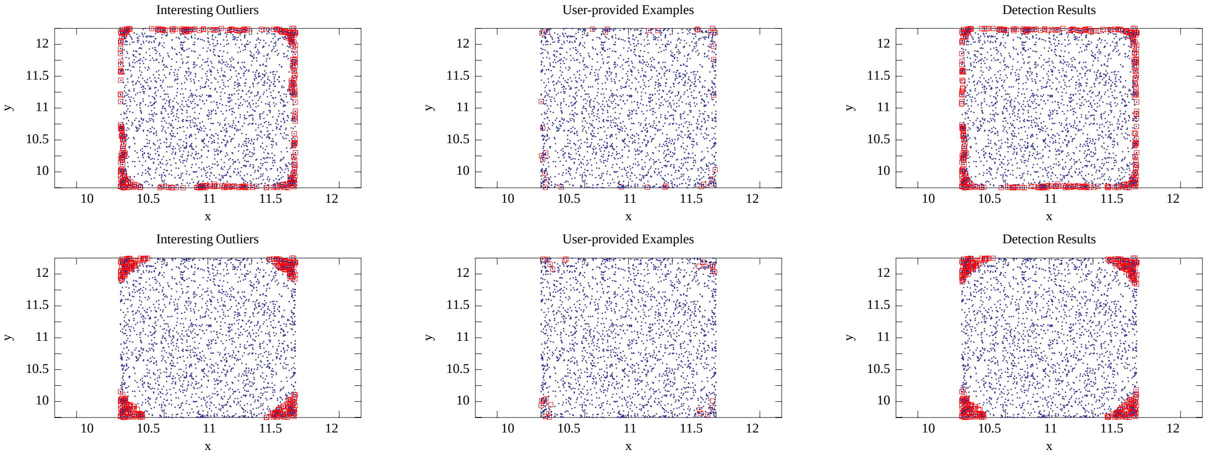

Figure 4　Detection Results on the Uniform Dataset. Top row: case U-Fringe, bottom row: case U-Corner—see Table 2 for description of each case.

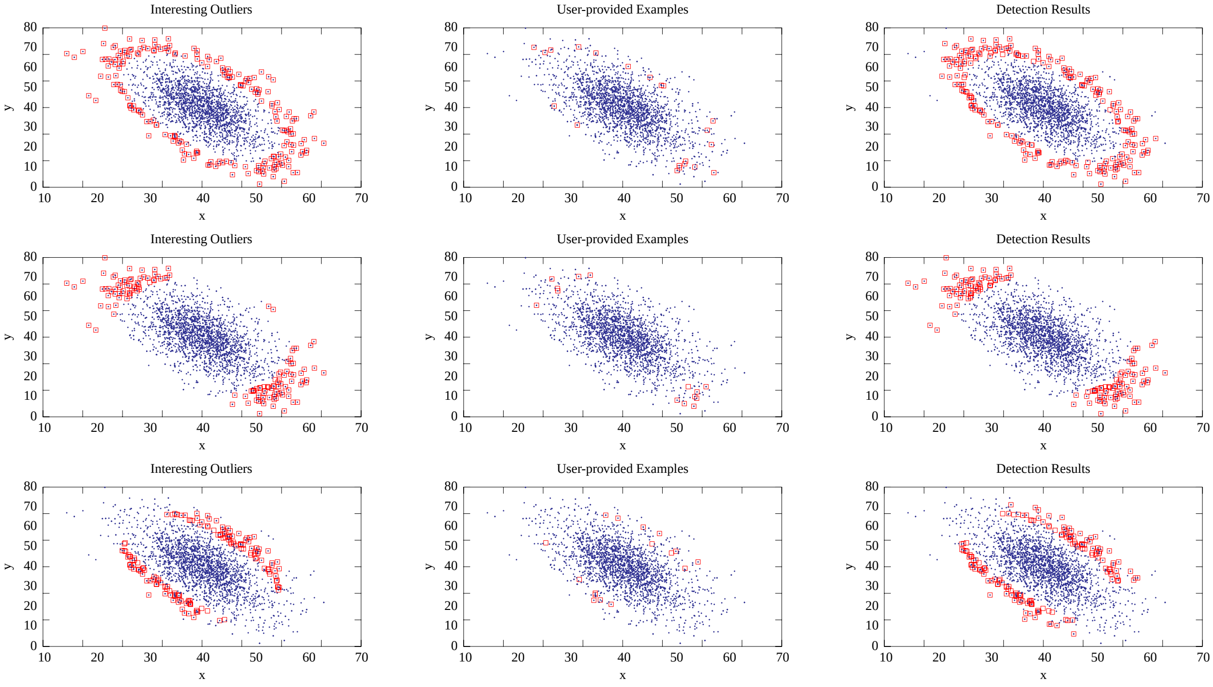

Figure 5　Detection Results on the Ellipse dataset. From top to bottom, in turn: case E-Fringe, case E-Long, case E-Short—see Table 2 for description of each case.

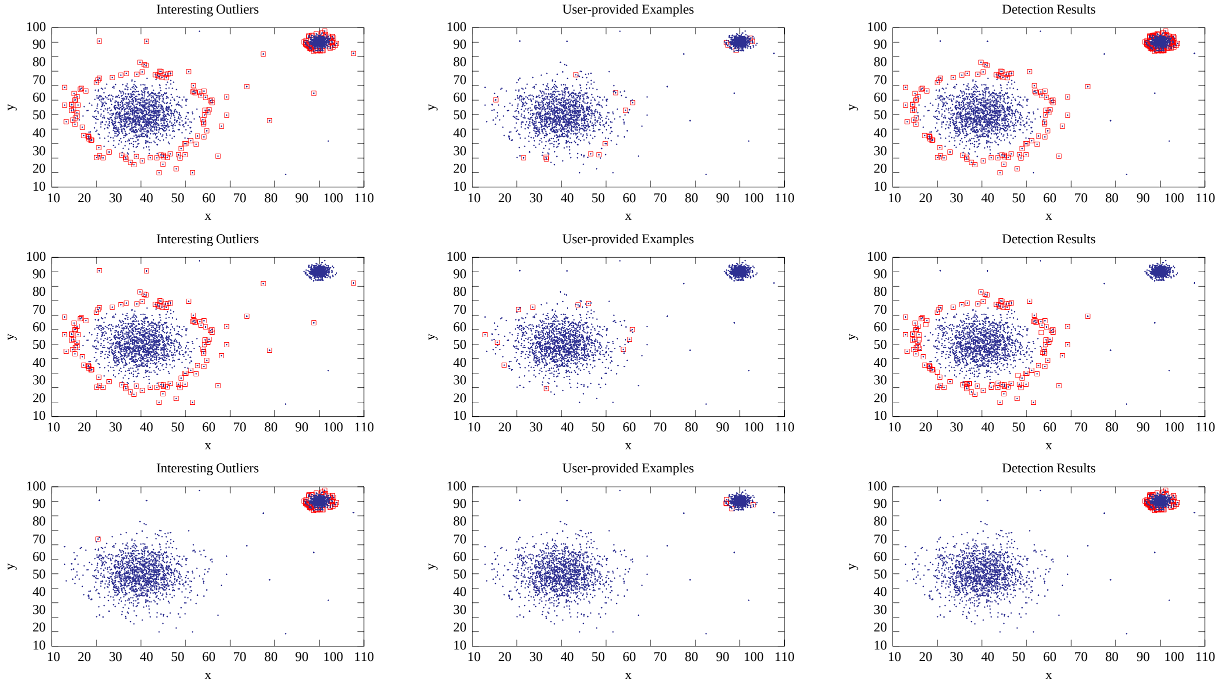

Figure 6 Detection results on the Mixture dataset. From top to bottom, in turn: case M-All, case M-Large, Case M-Small—see Table 2 for description of each case.

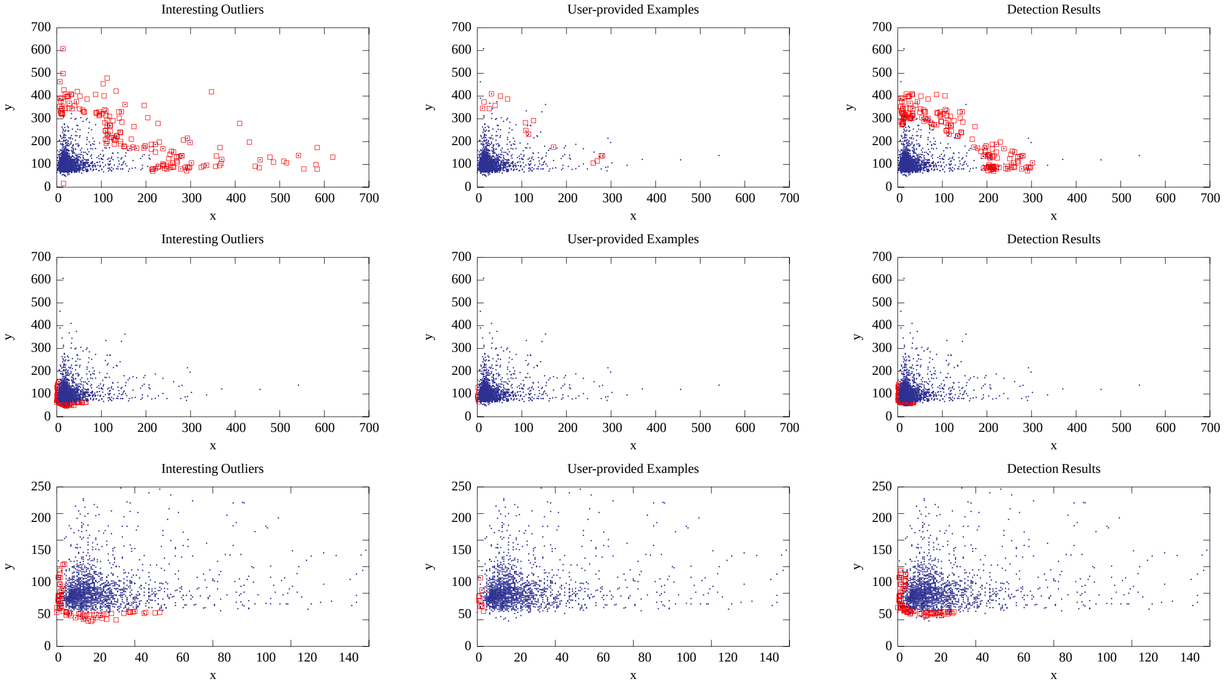

Figure 7 Detection Results on the Medical Dataset. From top to bottom in turn: Case M-Sector, Case M-Origin, A Zoom-In Version of Case M-Origin—see Table 2 for description of each case.

Table 3 Precision, recall showing performance of the proposed method. The number of iterations for convergence in the classification step is also shown.

Test Data		Precision	Recall	Iterations
Uniform Dataset	U-Fringe	82.76	88.18	8.1
	U-Corner	91.90	97.92	4.1
Ellipse Dataset	E-Fringe	84.02	89.77	4.6
	E-Long	95.97	97.30	5.7
	E-Short	83.26	89.94	6.7
Mixture Dataset	M-All	86.81	93.09	4.1
	M-Large	89.13	93.60	4
	M-Small	79.43	90.82	5.1
Medical Dataset	M-Sector	59.02	71.72	7.5
	M-Origin	62.70	77.33	5

examples in the discovery process and detect outliers with “outlierness” characteristics very similar to the given examples.

Acknowledgements

This research has been supported in part by Japan-U.S. Cooperative Science Program of JSPS and the Grant-in-Aid for Scientific Research from JSPS and MEXT (#15300027, #15017207).

References

[1] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. ACM Comp. Surveys, 31(3);264-323,1999.

[2] D.M. Hawkins. Identification of Outliers. Chapman and Hall, 1980.

[3] E.M. Knorr and R.T. Ng. Algorithms for mining distance-based outliers in large datasets. In Proc. VLDB 1998, pages 392-403, 1998.

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

[5] H. Yu, J. Han and k.Chang PEBL: Positive Example Based Learning for Web Page Classification Using SVM. SIGKDD, 2002

[6] http://lisp.vse.cz/pkdd99/Challenge/chall.htm

[7] http://www.csie.nut.edu.tw/ cjlin/libsvm

[8] K. Yamanishi, J.Takeuchi. Discovering Outlier Filtering Rules from Unlabeled Data. In ACM 2001, 1-58113-391-x /01/08

[9] M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In Proc. SIGMOD Conf., pages 93-104,2000.

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

[11] S. Papadimitriou, H. Kitagawa, P.B. Gibbons and C. Faloutsos. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In Proc. ICDE, pages 315-326, 2003.

[12] S. D. Bay and M. Schwabacher. Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. In SIGKDD’03, August 24-27, 2003.

[13] T.Johnson, I. Kwok, and R.T. Ng. Fast computation of 2-dimensional depth contours. In Proc. KDD, pages 224-228, 1998.

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