Interactive Outlier Detection Adaptive to Users' Intentions

Cui Zhu[†], Hiroyuki Kitagawa[†], Spiros Papadimitriou[††], Christos Faloutsos[††]

Abstract

Detecting outliers is an important, but tricky problem, since the preference of outlier-ness often depends on the user and/or the dataset. We have developed a system of detecting outliers that match users' intentions implied by outlier examples in prior work. In this paper, we propose a new refined method of interactive outlier detection adaptive to users' intentions.

Keywords Data Mining, Outlier Detection, Outlier Examples, User Interaction

1 Introduction

In applications like fraud detection, medical analysis, etc. outlier detection is an importent problem, since the rare events or exceptional cases are more interesting and useful than the common cases.

In the context of outlier detection, what makes the problem more difficult is that not everyone has the same idea of what constitutes an outlier. 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.

In most circumstances, users are experts in their problem domain and not in outlier detection. 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.

We have presented in our previous work [16, 17] a novel method that detects outliers adaptive to users' intensions implied by the outlier examples. In the method, users are intended to provide some examples and also the fraction of outliers they want to be detected.

However, it is hard for users to suppose the fraction of outliers in advance. To address the problem, we improve the system by interactive detection of outliers eliminating the need of the outlier fraction.

The remainder of the paper is organized as follows: In section 2, we discuss related work on outlier detection. In section 3, we introduce briefly the method of outlier detection based on examples. Section 4 presents the improved system. Section 5 reports the experimental evaluation on both synthetic and real dataset. Finally, Section 6 concludes the paper.

[†]Graduate School of Systems and Information Engineering, University of Tsukuba

[††]School of Computer Science, Carnegie Mellon University

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*, [15, 11]. *(2) Depth-based approach*, [14]. *(3) Clustering approach*, [1]. *(4) Distance-based approach*, [4, 5, 13]. 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*, [10]. 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 [12]. 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 by the system.

In order to deal with the curse of high dimensionality, a quite different technique is proposed in [2], where outliers are found by studying the behavior of projections from the data set. The most sparse *low − dimensional* cubes in the data are found by GA algorithm, and all the objects in these cubes are reported as outliers.

Another outlier detection method was developed in [9], 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, none of the existing methods can directly incorporate user in the discovery process. We have proposed to detect outliers using user provided examples [16, 17]. In this paper, we improve the method by interactive detection of outliers.

3 Outlier Detection by Examples

3.1 Measuring 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) [12] 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 of $n(p, \alpha, r)$, over all objects p in the r-neighborhood of p_i.

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

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

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. In our paper, the MDEF values are examined (or, sampled) at a wide range of sampling radii r, $r_{min} \leq r \leq r_{max}$.

[1]In all experiments, $\alpha = 0.5$ as in [12].

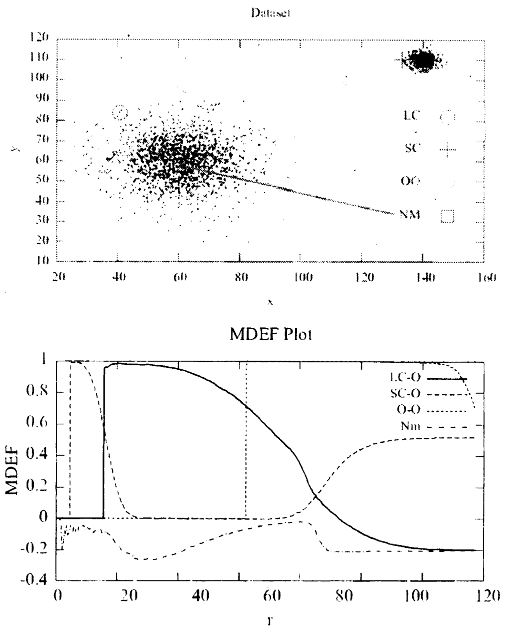

Figure 1: Illustrative Dataset and MDEF Plots.

To better illustrate MDEF, we give some examples. Figure 1 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.

From Figure 1, we can see that: (1) The MDEF values of normal objects, for example, the point named NM (box dot), keep low at *all* scales; (2) while outliers with dissimilar density distribution exhibit "outlier-ness" at different scales, or have different peaks of MDEF plots.

3.2 Detecting Outliers by Examples

Below we give a consice overview of the method according to [17]. The 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.

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 $m_{ij} = MDEF(p_i, r_j, \alpha r)$, $0 \leq j \leq n$, $r_0 = r_{min}$, $r_n = r_{max}$, $r_j = \frac{r_n - r_0}{n} j + r_0$.

Classification Step 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. Concretely, the algorithm uses the marginal property of SVMs to learn from the examples and the unlabeled data (i.e., the rest of the objects in the dataset).

The classification step consists of the following five sub-steps. *(1) 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. *(2) Training sub-step.* Train a SVM classifier using POS and NEG. *(3) Testing sub-step.* Use the SVM to divide the dataset into the positive set P and negative set N. *(4) Update sub-step.* Replace NEG with N, the negative data obtained in the testing sub-step. *(5) 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 2 summarizes the overall procedure of the method.

```
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 2: The Overall Procedure of the Method of Detecting Outliers by Examples

4 Proposed Method

The previously proposed method has the ability to detect outliers adaptive to users' intensions, given outlier examples.Aside from these examples, the algorithm uses as input the fraction of outliers to be determined. The fraction is used to determine the terminal condition of iteretions in the classification step.

We assume that the fraction of outliers is hard to figure out in prior. Here, we furthor develop the method to eliminate the need to input outlier fraction.

4.1 Modification of the Former Method

Without the information of fraction of outliers, the former terminal condition of iterations in classification step should be modified. Now we do iterations from the training sub-step to the update sup-step until the set of classified outliers P is same as P'. P' is the set of outliers in the previous iteration.

Accordingly, at the end of iteration, the hyperplane for separating outliers and normal objects is set as close as possible to the set of given examples. Therefore, the detected objects are those displaying stronger "outlier-ness" characteristics than the examples.

4.2 Proposed Method

In order to discover more outliers adaptive to users' intentions, we integrate the aforementioned modified detection method with a relevance feedback technique which enables the user to refine their preference by specifying relevant and non-relevant outliers (or normal objects). Based on users' feedback information, the system attempts to "guess" what are his intentions of outliers, how strong the "outlier-ness" they want.

The framework of interactive detection is as follows:

Classification Step Use the abovementioned classification method to set the hyperplane for separating outliers and normal objects as close as possible to the set of given examples.

Feedback Step In order to provide valuable feedback, i.e. raise questions of promising user outliers, we select randomly objects within the negative margin of the hyperplane.Even though, these objects have been classified as "normal" by the SVM in the classification step, they still exhibit "outlier-ness" characteristics close to the examples. The MDEF plots of the feedback objects are also presented to help users to determine whether the objects are interesting outliers. The "yes" answers, or positive feedbacks, are new outlier examples, and are incorporated into the positive training data therefor.

Convergence Step Iterate the classification and feedback steps until all the answers to feedbacks are "no". The hyperplane constructed at the end of the process will be close to the desired hyperplane which divides all the interesting outliers from normal objects. The objects separated by the last hyperplane are reported to the user as all detected outliers.

Figure 3 shows the flow of the proposed outlier detection method.

```
Input:
  Set of outlier examples: $E$
  Dataset: $D$

Output:
  Outliers like examples

Algorithm:
  $Q := \emptyset$  // Promising outliers
  $Q\_O := \emptyset$  // Outlier answers
  $Q\_N := \emptyset$  // Normal answers
  // Feature extraction step:
  For each $p_i \in D$
    For each $j$ $(0 \leq j \leq n)$
      Compute MDEF value $m_{ij}$
  // Convergence step:
  Do {
    $POS := E \cup Q\_O$
    // Classification step:
    $NEG$ := strongest negatives
    $P := D$
    Do {
      $P' := P$
      SVM := construct_SVM $(POS, NEG)$
      $(P, N)$ := SVM.classify $(D)$
      $NEG := N$
    } while $(|P| < |P'|)$
    // Feedback step:
    $Q$ := SVM.n_margin $(D)$
    $(Q\_O, Q\_N)$ := feedback $(Q)$
  } while $(|Q\_O| > 0)$
  return $P'$
```

Figure 3: 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.

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.[†] and "hide" the remainders.

3. Next, we detect outliers using the proposed method.
4. Finally, we compare the detected outliers with the (known) simulated set of interesting outliers. Precision (P), recall (R), and F1-measure (F1) defined below are adopted to measure the performance. F1-measure is a trade-off of precision and recall. A larger value of F1-measure indicates better performance.

$$P = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ predictions} \quad (2)$$

$$R = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ data} \quad (3)$$

$$F1 = \frac{2 * R * P}{(R + P)} \quad (4)$$

We use the LIBSVM [8] implementation for our SVM classifier. In all experiments, we use polynomial kernels and the same SVM parameters[‡]. Therefore, the whole processes can be done automatically.

For feedback, each time we select randomly z objects among the negative margin as questions. [§] "Yes" answers are given to the objects which are among the simulated interesting outliers. "No" answers are given to those not in the set of interesting outliers. We terminate the convergence step when the number of "yes" answers is zero.

5.2 Datasets and Sets of Interesting Outliers

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

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.

[†]In all experiments, $y = 10$.

[‡]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 $(u' * v + 1)^2$.

[§]In all experiments, $z = 4$.

Table 1: Description of Synthetic and Real Datasets.

Dataset	Description
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 [7], 7950 GPT and GLU examinations of patients in Chiba University hospital.

5.3 Experimental Results

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 4. The proposed method discovers both the underlying outlier notion and fraction!

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 5.

Mixture dataset Here we also mimic three cate-

Table 2: Interesting Outliers and the Discriminants.

Dataset	Cases			
	Label	Discription	Condition	# of Interesting Outliers
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

gories 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. Due to space constrains, the detection results of mixture dataset are not shown. The precition, recall and F1-measure results in Table 3 show that the proposed interactive method can almost perfectly detect outliers adaptive to these various intensions without the inputs of outlier fractions.

For all datasets, Table 3 shows the precision, recall and F1-measure metrics for the two methods. In the Table 3, the the former method is the method that detects outliers with inputs of fractions of outleirs, the improved method refers to interactive detection of outliers without fraction factors inputted. To compare performance of the two methods, we always use the same sets of user examples. The results in Table 3 are average of five trials. It also shows the number of iterations needed to converge in the improved method.

In all cases, with a little sacrifice of recall measurement, the interactive method surpassed the former method in detecting interesting outliers so far as precision measurement is concerned. F1-measure metrics exceed those of the former method. Note also that the number of feedbacks is always small (less than 10).

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 and act interactively with the system. Experiments on both real and synthetic data demonstrate that the interactive method can succesfully incorporate these examples in the discovery process and detect all outliers with "outlier-ness" 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, #16016205).

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] Charu C. Aggarwal, Philip S. Yu. Outlier detection for high dimensional data. In Proc. SIGMOD Conf., 2001.

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

[4] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In Proc. VLDB, pages 392-403, 1998.

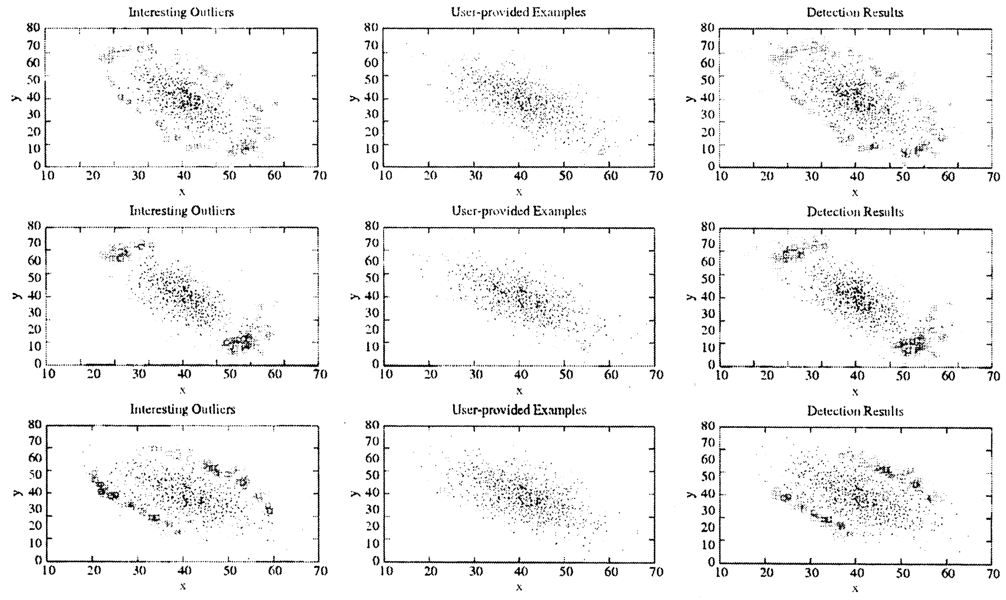

Figure 4: Detection Results on the Ellipse Dataset. From top to bottom, in turn: case E-Fringe, case E-Long, case E-Short—see Table 2 for description of each case.

[5] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-Based Outliers: Algorithms and Applications. VLDB Journal, 8:237-253, 2000.

[6] H. Yu, J. Han and K. Chang PEBL: Positive Example Based Learning for Web Page Classification Using SVM. In Proc. KDD, 2002.

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

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

[9] K. Yamanishi, J. Takeuchi. Discovering Outlier Filtering Rules from Unlabeled Data. In Proc. KDD, 2001.

[10] 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.

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

[12] 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.

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

[14] T. Johnson, I. Kwok, and R. T. Ng. Fast Computation of 2-Dimensional Depth Contours. In Proc. KDD, pages 224-228, 1998.

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

[16] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier by Example. In Proc. PAKDD, pages 222-234, 2004.

[17] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. Outlier Detection Adaptive to Users' Intentions. In Proc. Data Engineering Workshop (DEWS), 2004.

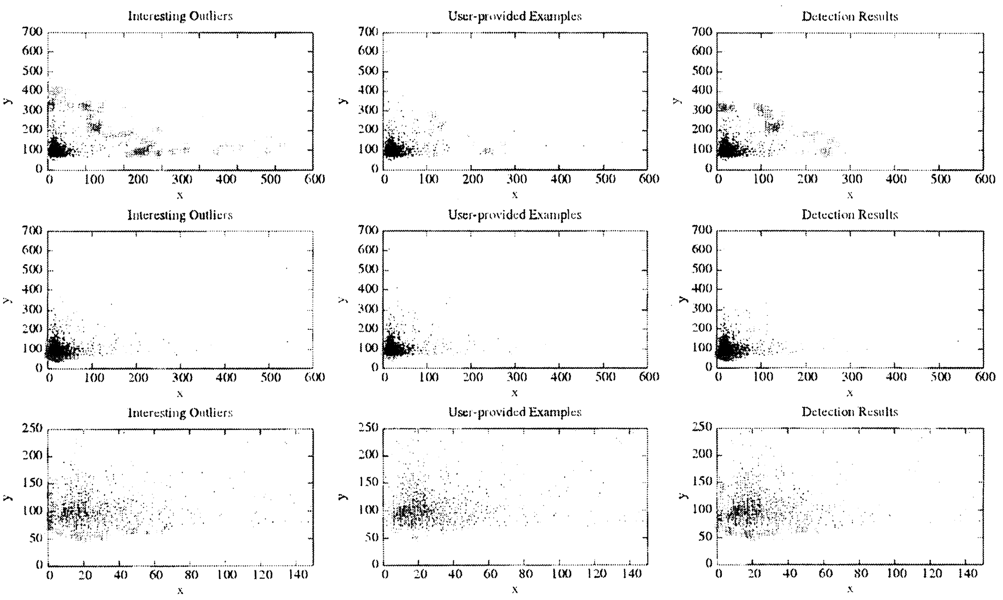

Figure 5: Detection Results on the Medical Dataset. From top to bottom in turn: Case M-Sector, Case M-Origin, A Zoom-In Version of Case M-Origin—see Table 2 for description of each case.

Table 3: Performance of the Former and Improved Method. Note that we alwayes use same sets of user examples to compare the two methods. Precision (P), recall (R) and F1-measure (F1) are used to show the performance. The number of iterations(Iter-) for convergence in the improved method is also shown.

Test Data		Former Method			Improved Method			
		P	R	F1	P	R	F1	Iter-
Ellipse Dataset	E-Fringe	84.02	89.77	0.8679	98.06	92.06	0.9497	10
	E-Long	95.97	97.30	0.9663	98.69	97.23	0.9795	4
	E-Short	83.26	89.94	0.8647	91.75	85.22	0.8836	4
Mixture Dataset	M-All	86.81	93.09	0.8984	93.61	89.88	0.9171	8.8
	M-Large	89.13	93.60	0.9131	98.00	94.74	0.9634	4.6
	M-Small	79.43	90.82	0.8474	92.87	82.86	0.8758	3.6
Medical Dataset	M-Sector	59.02	71.72	0.6475	94.61	57.91	0.7184	4
	M-Origin	62.70	77.33	0.6925	89.23	58.13	0.7040	3.8