J Intell Inf Syst (2011) 36:217–247
DOI 10.1007/s10844-010-0128-1

Outlier detection by example

Cui Zhu · Hiroyuki Kitagawa · Spiros Papadimitriou · Christos Faloutsos

Received: 9 February 2007 / Revised: 14 May 2010 / Accepted: 20 July 2010 /
Published online: 10 August 2010

Abstract Outlier detection is a useful technique in such areas as fraud detection, financial analysis and health monitoring. Many recent approaches detect outliers according to reasonable, pre-defined concepts of an outlier (e.g., distance-based, density-based, etc.). However, the definition of an outlier differs between users or even datasets. This paper presents a solution to this problem by including input from the users. Our OBE (Outlier By Example) system is the first that allows users to provide examples of outliers in low-dimensional datasets. By incorporating a small number of such examples, OBE can successfully develop an algorithm by which to identify further outliers based on their outlierness. Several algorithmic challenges and engineering decisions must be addressed in building such a system. We describe the key design decisions and algorithms in this paper. In order to interact with users having different degrees of domain knowledge, we develop two detection schemes:

C. Zhu (✉)
College of Computer Science, Beijing University of Technology,
Beijing, 100124, People's Republic of China
e-mail: cuizhu@bjut.edu.cn

H. Kitagawa
Graduate School of Systems and Information Engineering,
Center for Computational Sciences, University of Tsukuba,
Tsukuba, Ibaraki 305-8577, Japan
e-mail: kitagawa@cs.tsukuba.ac.jp

S. Papadimitriou
IBM T.J. Watson, Hawthorne, NY, USA
e-mail: spapadim@us.ibm.com

C. Faloutsos
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: christos@cs.cmu.edu

OBE-Fraction and OBE-RF. Our experiments on both real and synthetic datasets demonstrate that OBE can discover values that a user would consider outliers.

Keywords Outlier detection · Outlier example · Data mining · Machine learning

1 Introduction

Outlier detection in large datasets is one important procedure in data mining. It has many applications, including fraud detection, financial analysis, and health monitoring. Methods for finding rare events or "exceptional" objects in large datasets are increasingly drawing attention.

Different scientific communities define outliers differently, and such criteria can include outliers based on distribution (Barnett and Lewis 1994), distance (Knorr and Ng 1998), density (Breunig et al. 2000). Consequently, several approaches have been proposed. A fundamental issue, however, is that the notion of what an outlier is varies depending on the user, the problem domains, and even the datasets. For example, different users might have a different idea of what constitutes an outlier, the same user might view or define outliers differently based on how the data are arranged, and different datasets do not conform to any specific rules in all cases.

We look at objects that can be represented as low-dimensional, numerical tuples. Such datasets are prevalent in many applications. From a general perspective, as stated by Hawkins (1980) and Knorr and Ng (1997), an object is, intuitively, an outlier if it differs significantly from its neighbors. Of course, differing definitions of a neighborhood, what constitutes a difference, and whether or not the difference is significant will produce different sets of outliers. In the art of outlier detection, the most used measurements of difference are distance (Knorr and Ng 1998), density (Breunig et al. 2000), and their combination (Papadimitriou et al. 2003). The scale (i.e., radius) of the neighborhood is determined, implicitly or directly, by parameters that are supposed to be provided by users.

Example The following example might help to clarify the problem. Consider the dataset in Fig. 1. This dataset has a large sparse cluster, a small dense cluster, and some clearly isolated objects. By looking at the figure as a whole, only the circle dots appear to be outliers. In other words, when we examine wide-scale neighborhoods (i.e., by using a large radius, such as that covering nearly the entire dataset), only the isolated objects have very low neighborhood densities. However, the objects on the periphery of the large cluster (triangle dots) can also be regarded as outliers by using a smaller radius, as can the objects on the fringe of the small cluster (cross dots) by using a smaller radius. In this way, different objects are regarded as outliers depending on neighborhood scale or size.

This scenario is intuitive from the user's perspective. In most circumstances, users are experts in their problem domain but not in outlier detection. It is difficult for a user to decide the definition of an outlier before proceeding, but the user often has an idea or even examples of what would constitute an outlier. In the field of fraud detection, known cases of fraudulent activity or a novel form of fraudulent behavior are discovered by accident. These example outliers might describe the user's intentions, such as wanting to determine the characteristics of outliers similar

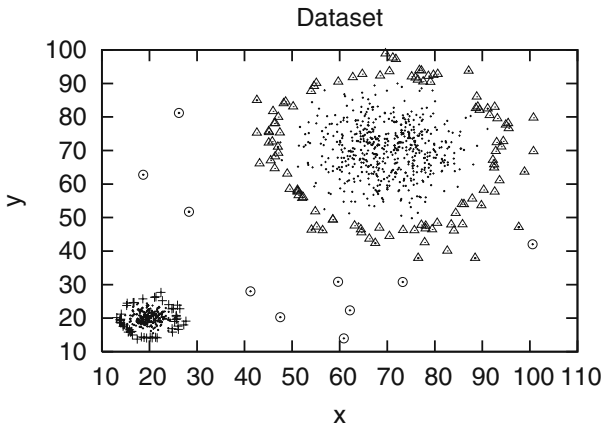


Fig. 1 Different kinds of outliers in a dataset

to the examples. Existing systems do not offer a direct way to incorporate such examples into the discovery process in a way that will discover more or future outliers.

This paper proposes Outlier By Example (OBE) method to do precisely that: to discover the so-called outlierness of any given point in a dataset at an appropriate scale, by using a small number of examples, and according to the distance- and density-based definition of an outlier. We are faced with several challenges in making this approach practical, the following being the most important. (1) What features best capture outlierness? These features should capture the important characteristics concisely and be efficiently computed. (2) The method should clearly require little user input and effectively use a small number of positive (i.e., outlier) examples. Further, it should not require negative (i.e., normal) examples. (3) Given these requirements, can we design methods to detect outliers by using only a handful of positive examples and unlabeled data? This paper describes the key algorithmic challenges and design decisions in detail.

We make the following contributions in this paper: (1) We introduce example-based outlier detection. (2) We demonstrate its intuitiveness and its feasibility. (3) We propose the OBE method, which is the first method to provide a solution to this problem. In order to deal with the different degrees of the users' domain knowledge, we have developed two schemes for the use of OBE. (4) We evaluate OBE by using both real and synthetic data with several small sets of outlier examples provided by users. Our experiments demonstrate that OBE can successfully incorporate these examples in the discovery process and detect outliers based on their outlierness characteristics.

Preliminary results have been reported by Zhu et al. (2004). However, this paper includes an analysis of OBE and proposes a new scheme (the OBE-RF), which detects outliers by using relevance feedback but which does not require the user to define the outlier fraction parameter. This paper also compares the OBE-RF with the OBE-fraction, which was proposed by Zhu et al. (2004). In addition, further experiments and observations regarding OBE's properties are presented.

The remainder of the paper is organized as follows: Section 2 discusses related work on outlier detection. Section 3 discusses the measurement of outlierness and

the different properties of outliers. Section 4 details the OBE framework and its two schemes. Section 5 reports the extensive experimental evaluation of both synthetic and real datasets. We further outline several issues regarding the application of OBE in Section 6. Section 7 concludes the paper.

2 Related work

In essence, outlier detection techniques traditionally employ unsupervised learning processes. Several existing approaches can be broadly classified into the following categories:

(1) *Distribution-based approaches* These are the classical methods used in statistics (Rousseeuw and Leroy 1987; Barnett and Lewis 1994). The user uses a statistical distribution to model the data points. Then, points that deviate from the model are flagged as outliers. However, these approaches are unsuitable for moderately high-dimensional datasets. It is difficult to determine which, if any, model fits an arbitrary dataset without prior knowledge of the data distribution.

(2) *Depth-based approaches* This computes the different layers of k-d convex hulls and flags objects in the outer layer as outliers (Johnson et al. 1998). It avoids the requirement of fitting a distribution to the data, but still suffers from the dimensionality curse.

(3) *Clustering approaches* Many clustering algorithms detect outliers as by-products (Jain et al. 1999). Since the main objective of these methods is clustering, they are not optimized for outlier detection.

(4) *Distance-based approaches* Distance-based outliers were originally proposed by Knorr and Ng in several papers from Knorr and Ng (1997) to Knorr and Tucakov (2000), and was improved by Ramaswamy et al. (2000) and Angiulli and Pizzuti (2005). Knorr and Ng defined a DB(β, λ)-outlier as a point p if at least β points in the dataset are further than λ from p. However, λ is difficult to determine and the method is sensitive to changes in λ. Another intuitive definition of distance-based outlier that can rank detected outliers was introduced by Ramaswamy et al. (2000). Given k and n, the top n points that have the maximum distances from their kth nearest neighbors are considered outliers. By using the same parameters of k and n, Angiulli and Pizzuti (2005) modified the definition of outlier by using the sum of the distances to the kth nearest neighbors as a measure of isolation. In general, the three definitions are based on a single, global criterion in terms of parameters β and λ or k and n. Thus, as pointed out in Breunig et al. (2000), these methods cannot cope with datasets having both dense and sparse regions. This is referred to as the multi-density problem.

(5) *Density-based approaches* To avoid the multi-density problem, Breunig et al. (2000) introduced a local outlier factor (LOF) for each object, indicating its degree of outlierness. LOF depends on the local density of its neighborhood, where the neighborhood is defined as the distance to the $MinPts^{\text{th}}$ nearest neighbor. However, LOF fails to deal with the multi-granularity problem. When there are clusters of various numbers of points, the method is unexpectedly sensitive to the $MinPts$ value (Papadimitriou et al. 2003).

(6) *LOCI* The multi-granularity deviation factor (MDEF), proposed by Papadimitriou et al. (2003), can cope with the multi-density and multi-granularity problems successfully. MDEF measures the outlierness of objects in neighborhoods of a defined scale. Their algorithm, LOCI, examines the MDEF values of objects in all ranges and flags as outliers those objects whose MDEF values deviate significantly from the local average in neighborhoods of the defined scale. Significant deviation is determined by using a threshold value of k_σ in their work. However, even though the definition of MDEF can capture outlierness in various scales, the user must examine these differences manually to identify outliers of personal interest.

A recently introduced method called StrOUD (Barbará et al. 2006), which is based on transductive confidence machines, processes every point in a new dataset separately and decides individually which point is an outlier according to an existing clustering model. To detect outliers in a dataset, it treats the data as both the training and the testing sets. When doing so, this method is similar to that proposed by Angiulli and Pizzuti (2005), which is based on distance. In Section 5, we show experimental comparisons between StrOUD, LOF, and the method proposed in this paper. The comparisons demonstrate the ability of our method to address the multi-density and multi-granularity problems and to detect outliers with varying degrees of outlierness by using user-supplied examples.

To deal with the curse of high dimensionality, a different technique was proposed by Aggarwal and Yu (2001), where outliers are found by studying the behavior of projections from the dataset. The most sparse low-dimensional cubes in the data are found by using a GA algorithm, and all objects in these cubes are reported as outliers. Based on this notion of outliers in sparse low-dimensional cubes, Zhu et al. (2005) proposed the method that seeks to find a subspace, where user examples are isolated from the majority. Objects that are also isolated in the same subspace are reported as outliers.

Another outlier detection method was developed by Yamanishi and Takeuchi (2001), in which a supervised learner and an unsupervised learner are combined. In their method, outliers are first detected by using SmartSifter (Yamanishi et al. 2000), which is based on a probabilistic model of the information source. Those detected outliers are then used to create labeled data for supervised learning of the outlier-filtering rules. Consequently, their labeled data are not provided by users. However, we detect outliers of interest by learning directly from user-provided examples.

In summary, most of all existing methods are designed to detect outliers based on prescribed criteria for outliers and require the users to set the parameters. To the best of our knowledge, ours is the first proposal for outlier detection in low-dimensional datasets by using user-provided examples.

3 Measuring outlierness

To understand user intentions and the outlierness of interest, the first step is to measure outlierness. It is crucial to select features that concisely capture the important characteristics.

As we have seen, the practical and popularly used definition of outliers in low-dimensional datasets is based on distance, density, or both. Among the several

measurements of outlierness (i.e., DB(β, λ)-outlier, LOF, and MDEF), MDEF can cope with both the multi-density and the multi-granularity problems. Therefore, we employ MDEF in OBE to measure the outlierness of objects in neighborhoods of different scales (i.e., radii).

A detailed definition of the multi-granularity deviation factor (MDEF) is given in Papadimitriou et al. (2003). Here we describe basic terms and notation. Let the r-neighborhood of an object p_i be the set of objects within distance[1] 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 or local neighborhood) and the r-neighborhood (sampling neighborhood) of p_i, respectively.[2] 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 and α, the multi-granularity deviation factor (MDEF) at radius (or scale) r is defined as follows:

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

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 this paper, MDEF values are examined (or sampled) over a wide range of sampling radii r, $r_{\min} \leq r \leq r_{\max}$, 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. For each p_i in the dataset, let $r_{\min,i}$ denote the distance to its nbth nearest neighbor. In our experiments, $r_{\min}$ is the minimum of all objects' $r_{\min,i}$. In other words, we do not examine the MDEF value of an object until the number of objects in its sampling neighborhood reaches nb,[3] which should be a reasonable number that effectively avoids the introduction of statistical errors in the MDEF estimates. We completed experiments by varying nb from 5 to 30 to determine $r_{\min}$, and found that the proposed method is insensitive to nb. In this paper, all experimental results are obtained when nb is 20.

Following are some examples that can illustrate MDEF. Figure 2 shows a dataset composed of two main groups: a large, sparse cluster and a small, dense one, both following a Gaussian distribution. There are also a few isolated points. Figure 2 also shows the MDEF plots for four objects in the dataset.

– Consider the point in the middle of the large cluster, NM, (at about $x = 70$, $y = 68$). The MDEF value is low at all scales: compared with its neighborhood, whatever the scale, the local neighborhood density is always similar to the average local density in its sampling neighborhood. Consequently, the object can always be regarded as a normal object in the dataset.

[1] The Euclidean distance measure is used to compute MDEF values in OBE.

[2] The value of α should be between 0 and 1. In all experiments, we set $\alpha = 0.5$ as in Papadimitriou et al. (2003).

[3] MDEF$(p_i, r, \alpha) = 0$ until $n(p_i, r) = nb$.

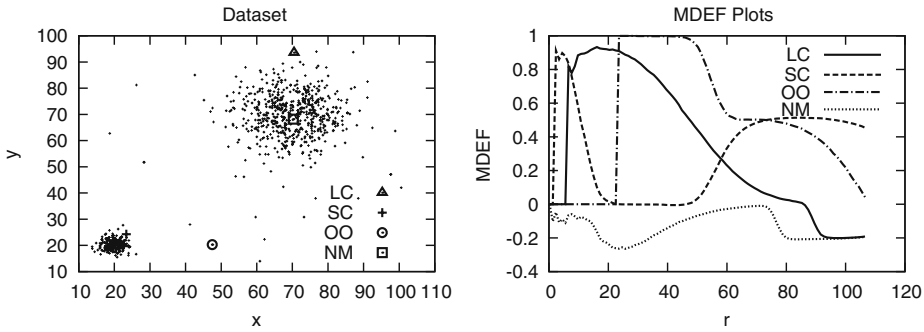


Fig. 2 Illustrative dataset and MDEF plots

- In contrast, for the other three objects (i.e., *LC*, *SC* and *OO*), there exist situations in which the MDEF values are very large, sometimes even approaching 1. This shows that they differ from their neighbors according to some scales. The greater the MDEF value, the stronger the degree of "outlier-ness."
  Even though the three objects in Fig. 2 can all be regarded as outliers, they still differ in that they exhibit outlierness at different scales.
- The MDEF value of the outlier in the small cluster, *SC*, (at about $x = 23, y = 24$), reaches its maximum at radius r of about 5. It then starts to decrease rapidly until reaching 0, where it remains at 23–45. The MDEF value then increases again but only to the degree of 0.6. This change in MDEF values indicates that the object is significantly abnormal compared with objects in the very small local neighborhood (objects in the small cluster).
- On the other hand, the outlier of the large cluster, *LC*, (at about $x = 70, y = 93$), exhibits strong outlierness in the range of $r = 10$ to 30, then becomes more and more ordinary as we take a broader view.
- For the isolated outlier, *OO*, (at about $x = 47$, $y = 20$), its MDEF value stays at 0 to almost $r = 22$, indicating that it is an isolated object. It then immediately displays a high degree of "outlier-ness."

4 Proposed method (OBE)

4.1 Overview

Our method of detecting outliers by example, denoted as OBE, is intended to be used for low-dimensional datasets, where the degree of outlierness is measured based on MDEF at various scales. OBE learns from user-provided outlier examples and discovers scales of interest to the users. Interesting outliers are detected as objects that display outlierness at the same scale, just as do the provided examples.

To this end, the first step is to extract an object's outlierness at different scales. This is accomplished by using the outlying feature-extraction step of OBE. Outlying-feature extraction, however, is only the beginning of the process. To detect outliers of interest, it is important to concentrate on a suitable neighborhood of interest. The selection of the neighborhood of interest is not trivial. OBE learns from outlier

examples to estimate the appropriate scales. The crucial point is how OBE can learn from a small number of outlier examples and not require negative data (i.e., normal objects). The outlier-detection step of OBE addresses the problem by using an iterative SVM classifier.

The following section provides an overview of the SVM classifier. Details of the OBE method and its two schemes are then explained.

4.2 Overview of SVM

As a binary classification algorithm, support vector machines (SVM) is a standard tool for machine learning and data mining. It has demonstrated outstanding performance in many domains of classification problems, such as text categorization (Joachims 1998), image classification (Goh et al. 2001), and biosequence analysis (Markowetz 2003). Interest is growing in SVM classifiers in the learning and application communities.

SVM possesses several important properties, the most important of which are listed below.

- Maximization of margin. This property can be illustrated by using a linear SVM, which is its simplest form. A linear SVM is a hyperplane that separates the two sets of positive and negative training data by using a maximum margin in the feature space. The margin (m) implies the distance from the hyperplane (i.e., class boundary) to the nearest positive and negative objects in the feature space. The objects closest to the separating hyperplane are called support vectors. An example of a simple problem that is linearly separable is shown in Fig. 3.
- The distance from the hyperplane to an object approximates the relative strength of the properties that distinguish positive objects from negative ones. For instance, a strong negative object should be located far from the class boundary on the negative side in the SVM feature space. Conversely, the positive or negative property of an object within the margin is ambiguous and the object is difficult to classify.
- Linear or nonlinear transformation of the input space to the feature space can be accomplished by using kernel methods. Advanced kernel methods, such as polynomial and Gaussian kernels, can be used to transform the input space to another high-dimensional feature space when the training data cannot be

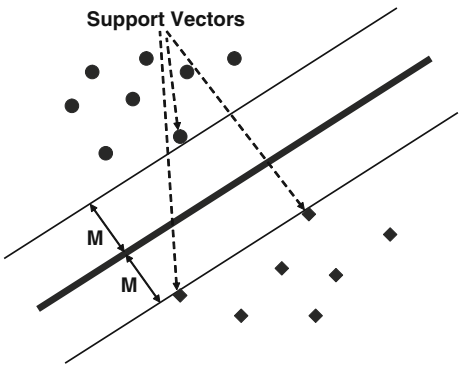


Fig. 3 An illustration of a linear SVM in a two-dimensional space

separated linearly. Linear kernels are fast, but nonlinear kernels have better accuracy for some specific problems (Tax and Duin 1999). In our experiments, polynomial kernels generally have superior performance. A discussion of the choice of kernels for outlier detection is given in Section 6.

– The optimal hyperplane is completely defined by using support vectors. That is, a hyperplane built from the entire set of training data is the same as that built from the picked support vectors only. This property is quite helpful where a plentiful supply of training data is difficult to obtain and where support vectors are all that are available. Other learning methods, such as probabilistic methods (e.g., Fisher's linear discriminant analysis) or decision trees, do not possess this advantage and potentially require sufficient training data.

4.3 Outlying feature extraction step

The purpose of this step is to map all objects from the input space into the MDEF-based feature space, where the MDEF plots of objects capturing the degree of outlierness, as well as the scales at which the outlierness appears, are represented by vectors. Let D be the set of objects. In the MDEF-based feature space, each object is represented by a vector: $O_i = (m_{i0}, m_{i1}, \ldots, m_{in})$, $O_i \in D$, where $m_{ij} = \text{MDEF}(p_i, r_j, \alpha)$, $0 \le j \le n$, $r_0 = \min\{r_{\min,k} \mid p_k \in D\}$, and $r_n = r_{\max}$, $r_j = \frac{r_n - r_0}{n} j + r_0$.[4] Here, n denotes the number of sampling radii.[5]

In the context of outlier detection, there might exist some objects that are such strong outliers that they might be highlighted by all users (i.e., the circle objects in Fig. 1, which nearly all users would consider to be outliers). We call these outstanding outliers. After all objects are projected onto the MDEF-based feature space, we can discover outstanding outliers.

Outstanding outliers The set of outstanding outliers is defined by $\{O_i | \max\_M(O_i) > k_\sigma,\ O_i \in D\}$, where $\max\_M(O_i) = \max\{m_{ij} \mid j = 0, 1, \cdots, n\}$ and k_σ is a threshold.

An outstanding outlier is one that exhibits an extremely high degree of outlierness at some scales, as measured by k_σ. From the definition of MDEF, k_σ should be less than 1 but should be large enough to identify accurately any outstanding outliers. A k_σ threshold that is not sufficiently high will result in some objects being mislabeled as outstanding outlying. When these mislabeled outstanding outliers are incorporated into positive training data in the following outlier-detection step, they might counteract the ability of the model to detect outliers at the desired scales. In all of our experiments, we set $k_\sigma = 0.99$, which is sufficiently high for most applications.

In the MDEF-based feature space, we can also filter out false outlier examples, if any, that have low MDEF values in all ranges. Based on the MDEF conditions, these objects do not show outlying features at any scale and are therefore not outliers.

We obtain the positive training data by combining the user-supplied outlier examples and the outstanding outliers.

[4] More precisely, if $r_j \ge r_{\min,i}$, then $m_{ij} = \text{MDEF}(p_i, r_j, \alpha)$, otherwise $m_{ij} = 0$.

[5] We completed experiments by varying the number of n from 50 to 400. The results showed that OBE is insensitive to n. In all our experiments of OBE, $n = 100$.

4.4 Outlier detection step

So far, the positive training data, as well as the entire, unlabeled dataset, are available to us. The next step is to find an efficient, effective algorithm to discover the hidden outlier concepts that the user has in mind.

We use an SVM classifier to learn the outlierness of interest to the user and then to detect outliers that match this outlierness in MDEF-based space, where the features of outlierness on all scales are characterized. Traditional classifier construction needs both positive and negative training data. However, it is too difficult and also a burden for users to provide negative or normal data for outlier detection. Most objects fall in the category of negative data and it is unreasonable to expect users to examine them.

However, OBE addresses this problem and can learn merely from the positive examples and the unlabeled data (i.e., the remaining objects in the dataset). The algorithm shown here uses the maximal margin feature of SVMs. In this sense, the algorithm generally resembles PEBL (Yu et al. 2002), which also learns 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 OBE, the positive examples are merely examples of outliers, and it is not desirable to set the hyperplane as in PEBL. Another difference between OBE and PEBL is that strong negative data are determined by taking the characteristics of MDEF into consideration.

To learn from the positive examples and unlabeled data, OBE constructs initial negative training data in the MDEF-based feature space by extracting the most-normal objects from the unlabeled data. Then, by using an iteration of learning and detection, the hyperplane for separating positive and negative objects is pushed progressively towards the desired position, where the separated positive objects are the outliers of interest. In OBE, the final separating hyperplane is decided by using two schemes to deal with differing degrees of domain knowledge. The first scheme (OBE-Fraction) sets the final hyperplane based on the fraction of the desired outliers. The second (OBE-RF) judges the position of the final hyperplane by using relevance feedback and interaction with the users.

4.4.1 Detection by using fraction parameters

Aside from the outlier examples, the scheme for detection outliers by using the fraction parameter (*OBE-Fraction*) uses as the input the fraction of outliers to be determined. The fraction is used to determine the iteration termination conditions in the detection step. OBE-Fraction consists of the following five sub-steps.

Strong negative data extraction All objects are sorted in descending order according to $\max\_M(O_i)$. The objects at the bottom of the list have low MDEF values at all scales. According to the characteristics of MDEF, these objects can always be regarded as normal objects in the dataset. Thus, from the objects at the bottom of the list, we select the same number of (strong) negative training data as the number of positive training data. Let the set of strong negative training data be NEG. Also, let the set of positive training data be POS.

Training Train an SVM classifier by using POS and NEG.

Testing Use the SVM to divide the dataset into a positive set P and a negative set N.

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

Iteration By iterating from the training sub-step to the updating sub-step, we obtain progressively more negative data and these negative data gradually push the SVM's hyperplane towards POS. When the ratio of classified positive objects in P converges to the fraction specified by the user, we terminate the iteration. The objects in the final P are reported to the user as detected outliers.

Figure 4 summarizes the overall procedure of the scheme.

4.4.2 Detection by using relevance feedback

In some cases the fraction of outliers is difficult to determine beforehand. We therefore developed a scheme called OBE-RF, which detects outliers without the need for prior input of the outlier fraction parameters. To discover more outliers in response to user intentions, we integrate the iteration of classification by using a relevance-feedback technique, which enables the user to refine preferences by specifying relevant and non-relevant outliers (or normal objects). Based on user feedback information, the system attempts to guess the user's intentions regarding outliers and the strength of outlier-ness the user wants.

The OBE-RF procedure is as follows:

Classification Starting with the positive and strong negative training data, conduct iterations from the training sub-step to the updating sub-step until the difference in the size of P between two iterations is less than ε. Accordingly, at the end of this phase the hyperplane for separating positive and negative objects is set as close as possible to the set of positive training data. Therefore, the objects classified as positive are those displaying stronger outlier-ness characteristics than the examples.

```
Input:
    Set of outlier examples: E
    Fraction of outliers: F
    Dataset: D
Output:
    Outlier-like examples
Algorithm:
    OO := ∅   // Outstanding outliers
    // Outlying feature-extraction step:
    For each p_i ∈ D
        For each j (0 ≤ j ≤ n)
            Compute MDEF value m_ij
            If m_ij > k_σ
            Then OO := OO ∪ {p_i}
POS := E ∪ OO
// Outlier-detection step:
NEG := strongest negatives
P := D
Do {
    P' := P
    SVM := train_SVM (POS, NEG)
    (P, N) := SVM.classify (D)
    NEG := N
} while (|P| ≥ F * |D| and |P| ≠ |P'|)
return P'
```

Fig. 4 Overall procedure for OBE-fraction

```
Input:
    # of queries per feedback: m
    Set of negative classified objects: N
Output:
    Feedback
Algorithm: feedback (m, N)
    Q_Y := ∅   // Outlier answers
    Q_N := ∅   // Normal answers
    F :=raise first query
    For i = 2 to m do
        If answer of F is yes, Then
            Q_Y := Q_Y ∪ {F}
            F :=raise next query by stepping forward
        Else
            Q_N := Q_N ∪ {F}
            F :=raise next query by stepping backward
    return Q_Y ∪ Q_N
```

Fig. 5 The feedback procedure in OBE-RF

Feedback To detect more outliers, it is desirable that some valuable queries, or promising outliers, be found and be determined by users as relevant or non-relevant outliers.

OBE chooses promising outliers at the negative side by using a heuristic method. First, the object near the middle of the negative margin is raised as a query. Although this object has been classified as negative by the SVM in the previous classification sub-step, it still exhibits some outlier-ness characteristics that are similar to those of the positive training data. The system then progressively selects the next object as a question to users based on the answers to previous queries. Specifically, if a yes or outlier answer is given, OBE proceeds to find the next question for potential outliers by taking an additional step, moving farther from the hyperplane on the negative side. The next question will determine the object whose distance from the hyperplane is nearest to the distance of the previous yes feedback plus S.[6] Conversely, if the answer is no, OBE steps back to select an object at the middle of the previous no-answered query and the previous yes-answered query (or the hyperplane itself when there are no yes queries). In this way, OBE raises a number m of queries by stepping forward or backward heuristically based on previous feedback. The feedback procedure is shown in Fig. 5.

The yes answers, or positive feedbacks, are new outlier examples, whereas objects receiving no answers are negative data. Thereafter, these new outlier examples and negative data are incorporated into the training data.

Convergence Iterate the classification and feedback sub-steps until the answers for all queries are no. The hyperplane constructed at the end of the iteration will be close

[6] S is the pace of stepping forward. For simplicity, we set $S = 1$ in our experiments.

to the desired hyperplane, which filters out all of the interesting outliers from the normal objects. The positive objects separated by the last hyperplane are reported to the user as all detected outliers.

Figure 6 shows the OBE-RF procedure.

4.5 Complexity of OBE

The algorithm of OBE's two schemes comprises procedures for computing MDEF values at sampling radii and iterations of the classification steps. In our experiments, the time required for the procedure in which the classification iteration in both schemes are accomplished is significantly less than that of the procedure used to compute MDEF. The complexity of computing MDEF is $O(N^2 \times (\log n + n))$, where N is the dataset size and n refers to the number of sampling radii. Therefore, $O(N^2 \log n)$ is used to allocate $O(N^2)$ distances into n ranges of radii and the computational cost of n-dimensional MDEF vectors for N objects is $O(N^2 n)$. Usually, the number of sampling radii (i.e., n), is significantly less than N.

4.6 Analysis of OBE

Fast convergence of the hyperplane of OBE Because no false-negative results are generated in the strong negative data extraction sub-step and the algorithm of SVM maximizes the margin in its feature space, the hyperplane of OBE approaches its final position. The number of iterations required for convergence in OBE is logarithmically related to the distance between the strong negatives and the final position in the SVM's high-dimensional feature space.

In the detection scheme employing fraction parameters, the final position is decided by the outlier fraction parameter. For the classification sub-step used by OBE-RF, the final position of the hyperplane is outside of but close to the positive

```
Input:
  Set of outlier examples: E
  # of queries per feedback: m
  Dataset: D
Output:
  Outlier-like examples
Algorithm:
  OO := ∅   // Outstanding outliers
  Q_Y := ∅   // Outlier answers
  Q_N := ∅   // Normal answers
  // Outlying feature-extraction step:
  For each p_i ∈ D
    For each j (0 ≤ j ≤ n)
      Compute MDEF value m_ij
      If m_ij > k_σ
      Then OO := OO ∪ {p_i}
POS := E ∪ OO
// Outlier-detection step:
Do {
  POS := POS ∪ Q_Y
  NEG := strongest negatives ∪ Q_N
  P := D
  Do {
    P' := P
    SVM := construct_SVM (POS, NEG)
    (P, N) := SVM.classify (D)
    NEG := N
  } while (|P'| − |P| > ε)
  (Q_Y, Q_N) := feedback (m, N)
} while (Q_Y ≠ ∅)
return P'
```

Fig. 6 Procedure for OBE-RF

training data, so the classified outliers display stronger outlierness than the positive training data.

Proof The intersection of the strong negative training data and the outlying objects (i.e., false negative) is null. This is because we extract the strong negative training data as the most-normal objects, which have the lowest MDEF values at all scales. Let NEG_0 denote the strong negative training data. Suppose the distance of NEG_0 to POS in the feature space of SVM is d_0. Trained from NEG_0 and POS, the SVM classifier sets the hyperplane between NEG_0 and POS, which maximizes the margins of positive and negative sides equally. Objects at the negative side of the hyperplane become new negative training data, NEG_1, and thus the distance d_1 of NEG_1 from POS is half that of d_0. The hyperplane of SVM trained from NEG_{i-1} and POS is set repeatedly between NEG_{i-1} and POS with equal margins. Thus, d_i is always half of d_{i-1}. Therefore, the hyperplane is pushed towards POS, and the number of iterations will be logarithmically related to the distance of NEG_0 from the final position in the SVM feature space. □

Convergence safety If NEG_0 does not include any false negatives, and the algorithm of SVM does not misclassify separable objects in POS (i.e., the positive training data), the hyperplane of SVM does not trespass on the space of POS, regardless of the number of iterations in OBE.

Proof Because OBE picks as NEG_0 those objects that have the lowest MDEF values at all scales, NEG_0 is the set of the most-normal objects. POS is the positive training data, that is, the true outlier examples and outstanding outliers. Therefore, $\mathrm{NEG}_0 \cap \mathrm{POS} = \emptyset$. $\mathrm{NEG}_{i+1} \cap \mathrm{POS} = \emptyset$, if $\mathrm{NEG}_i \cap \mathrm{POS} = \emptyset$. This is because NEG_{i+1} is the set of objects separated by a hyperplane, which is constructed from NEG_i and POS, and separable objects in POS are not misclassified as negatives. This is guaranteed in our implementation of the SVM classifier by setting the parameter C (the penalty imposed on the training data that fall on the wrong side of the decision boundary) as 1,000 (i.e., a very high penalty for misclassification).[7] Therefore, the hyperplane of SVM does not trespass on POS, regardless of the number of iterations in OBE. □

For many scenarios, it is hard to say whether an object is an outlier or not. Users are likely to argue that a particular object is more of an outlier than are others. In OBE-Fraction, the fraction parameter is used to imply the degree of outlierness, that is, the most-outlying objects at the scales of concern are regarded as outliers. Therefore, we suppose the parameter of the outlier fraction is specified as a reasonable value, so the final hyperplane decided by the fraction is outside the space of POS and more outliers of interest are discovered. If users set the fraction parameter too small (e.g., the space of the outliers decided by the fraction is less than that of POS), the hyperplane converges to the POS boundary rather than to that decided by the fraction. This occurs because it does not trespass on POS.

In the classification sub-step of OBE-RF, the hyperplane is pushed towards the positive training data POS and is located outside the space of POS. This implies that

[7] In fact, in all of our experiments, instances in which the outlier examples or outstanding outliers are misclassified as negatives were never observed.

hyperplane accuracy depends on the quality of the positive training data. Within the context of outlier detection, positive data tend to be undersampled because, usually, only a few outlier examples are available. Thus, as observed in our experiments, the final hyperplane in the classification sub-step is often adjacent to the space of the outlier examples. The unlabeled objects that are outside the space of the positive training data but are true outliers will be classified as normal by the final hyperplane in the classification sub-step. In that case, it is better to set the hyperplane loosely around the positive training data. We therefore set $\varepsilon = |D|/1{,}000$ in OBE-RF for a loosely separating the hyperplane in all of our experiments.

4.7 Limitation of OBE

The OBE framework can essentially work based on any definition of outliers that estimates the degree of outlierness at different scales or in different views. As stated in Section 2, most existing methods measure the difference in objects based on such parameters as distance (Knorr and Ng 1998), density (Breunig et al. 2000), and their combinations, as is done in MDEF (Papadimitriou et al. 2003). Our OBE framework uses MDEF because MDEF can cope successfully with the multi-density and multi-granularity problems. These proximity-based definitions of an outlier are meaningful in low-dimensional datasets (Knorr and Ng 1999). However, when the dimensionality is very high and the data are sparse, it is hard to tell which one is an outlier from the standpoint of proximity (Beyer et al. 1999). Meaningful outliers are more likely to be defined by examining the behavior of the data in low-dimensional projections (Aggarwal and Yu 2001). It is interesting to study detection of proximity-based outliers in projections of high-dimensional datasets.

5 Experimental evaluation

This section describes our experimental methodology and the results obtained by applying OBE to both synthetic and real data, which further illustrates inherent intuition and also demonstrates the effectiveness of our method.

5.1 Datasets

We use two synthetic and two real datasets (Table 1) to evaluate OBE.

5.2 Experimental procedure

Our experimental procedure is as follows:

1. To simulate interesting outliers, we start by selecting objects that represent outlierness at some scale. Specifically, we use discriminants of the form $\bigwedge_q(\min_q, \max_q, \mathrm{Cond}_q, K_q)$, where $(\min_q, \max_q, \mathrm{Cond}_q, K_q)$ stands for the condition that $(m_{ij}\ \mathrm{Cond}_q\ K_q)$ holds for some j, such that $\min_q \le j \le \max_q$ and Cond_q can be either $>$ or $<$. K_q is a threshold to define the degree of outlierness, and m_{ij} denotes the MDEF value of object i at radius j. The left and right boundaries of the interesting rand are denoted by $\min_q$ and $\max_q$, respectively.

Table 1 Description of synthetic and real datasets

Dataset	Dimensionality	Description
Ellipse	2	A 6,000-point ellipse following a Gaussian distribution
Mixture	2	A 5,000-point sparse Gaussian cluster, a 2,000-point dense Gaussian cluster, and 10 randomly scattered outliers
NYWomen	4	Marathon runner data, 2,229 women from the NYC marathon: average pace (in minutes per mile) for each stretch (6.2, 6.9, 6.9, and 6.2 miles)
Abalone	3	Abalone data, obtained from the UCI (http://www.ics.uci.edu/~mlearn/MLRepository.html) machine learning repository, 4,177 diameter, whole weight, and ring examinations of abalones

2. We then hide most of these outliers. In particular, we randomly sample y% of the outliers to serve as examples that would be picked by a user.
3. Next, we detect outliers by using the OBE-Fraction and OBE-RF schemes separately.
4. Finally, we compare the detected outliers to the simulated (target) set of outliers. More specifically, we evaluate the success of the two OBE schemes in recovering the hidden outlier concept by using precision/recall/F1 measurements.

OBE reports as interesting those outliers that are outstanding, as well as those returned by the classifier. 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 example, the discriminant (1, 35, >, 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 35. We always randomly sample 10% ($y = 10$) of the interesting outliers to serve as user-provided outlier examples and hide the rest. The number of examples is shown in the right-most column of Table 2.

To detect outstanding outliers, we use $k_\sigma = 0.99$ for the complete synthetic and real datasets. The number of discovered outstanding outliers is shown under the heading "#_OOutliers" in Table 2.

For the OBE-RF scheme, we conduct the iteration in the classification sub-steps until the difference in size of P between two iterations is less than the data size divided by 1,000 ($\varepsilon = |D|/1{,}000$). Thus, a loosely separating hyperplane is determined as discussed in Section 4.6. We always raise four queries of objects ($m = 4$) at the negative side of the hyperplane in each feedback sub-step, and terminate the detection process when all answers are no.

We use the LIBSVM implementation (by Chang and Lin, http://www.csie.nut.edu.tw/~cjlin/libsvm) for our SVM classifier. We extensively compared the accuracy of the SVM kernel on a linear, polynomial, and radial basis and found that polynomial kernels consistently perform better, as described in Section 6. Therefore, here we present the results of using polynomial kernels and the same SVM parameters.[8] This makes it possible for all processes to be done automatically.

[8]For the polynomial kernel, we use a kernel function of $(u' * v + 1)^2$.

Table 2 Interesting outliers and their discriminants

Dataset	#_OOutliers	Cases				
		Label	Description	Condition	#_IOutliers	#_Examples
Ellipse dataset	4	E-F	Fringe	(5, 30, >, 0.85)	209	21
		E-L	Long ends	(15, 25, >, 0.8) (30, 40, >, 0.6)	140	14
		E-S	Short ends	(5, 15, >, 0.8) (35, 40, <, 0.6)	162	16
Mixture dataset	13	M-A	All	(1, 35, >, 0.9)	163	16
		M-L	Large cluster	(15, 35, >, 0.9)	117	12
		M-S	Small cluster	(1, 5, >, 0.9)	58	6
NYWomen dataset	16	N-FS	Very fast or slow	(800, 1400, >, 0.7)	89	9
		N-PF	Partly fast	(300, 500, >, 0.8) (1,400, 1,600, <, 0.4)	123	12
		N-SS	Stable speed	(100, 300, >, 0.8) (400, 600, <, 0.3)	104	10
Abalone dataset	8	A-FS	Very fat/slim	(0.1, 0.2, >, 0.9)	97	10
		A-HA	Huge/aged	(0.4, 1, >, 0.9)	68	7

#_OOutliers and #_IOutliers are numbers of outstanding outliers and interesting outliers, respectively. Examples are outlier objects that are supposed to be provided by users

We report the effectiveness of OBE in discovering the hidden outliers by using precision, recall, and F1 measurements:

$$\text{Precision} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}} \tag{2}$$

$$\text{Recall} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive data}} \tag{3}$$

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}. \tag{4}$$

5.3 Experimental comparisons

Experimental comparisons of our OBE technique against StrOUD and LOF are performed on all four datasets by using precision/recall/F1 measurements with respect to the target class of outliers only (i.e., for which the user has provided samples). To find the nearest neighbors, we test a large range of choices[9] for the parameters *MinPts* in LOF and *K* in StrOUD.

For each *MinPts* within the range of choices, we compute LOF values for all objects and rank them with respect to their LOF values. On the ranked list, the top #_*IOutliers* objects are flagged as LOF outliers. #_*IOutliers* is the number of interesting outliers for each case and is listed under the "#_IOutliers" column of

[9]The large range of choices is (2, 5, 10, 20, 30, 40, 50, 60, 80, 100, 150, 200, 250, 300, 400, 500, 700, and 1,000).

Table 2. That is, the number of positive predictions of LOF is exactly the same as that of positive data. Thus the precision, recall, and F1 measurements of the LOF method are identical. For each testing case, we select among the large range of choices the right *MinPts* for LOF, which results in the best F1 measurement, and the corresponding result is used for comparison with OBE.

For StrOUD, we suppose that there is no clustering information available and that the data are treated as a whole (as if it all belongs to one cluster). Thus, the required confidence level can be reflected directly by τ ($\delta = 1 - \tau$). To be comparable, we set the τ for each case respectively as the number of interesting outliers divided by the size of the dataset. By doing so, the number of outliers declared by StrOUD is almost the same as that of the target outliers, and the computed precision, recall, and F1 measures are almost equal. Also, by examining the large range of choices for K, the best results (i.e., those with the best F1 measures) of StrOUD are selected and compared with OBE.

To save space, only the comparison results between OBE, LOF, and StrOUD on the Ellipse dataset are visually presented in Fig. 7a, b, and c. The experimental results of all methods are summarized in Table 3.

5.4 Results

As Table 3 shows, the performances of the OBE-Fraction and OBE-RF schemes are nearly comparable and OBE-RF is a little better in most cases. Thus, the experimental results shown here are from OBE-RF schemes to demonstrate the ability of OBE to detect outliers based on user examples.

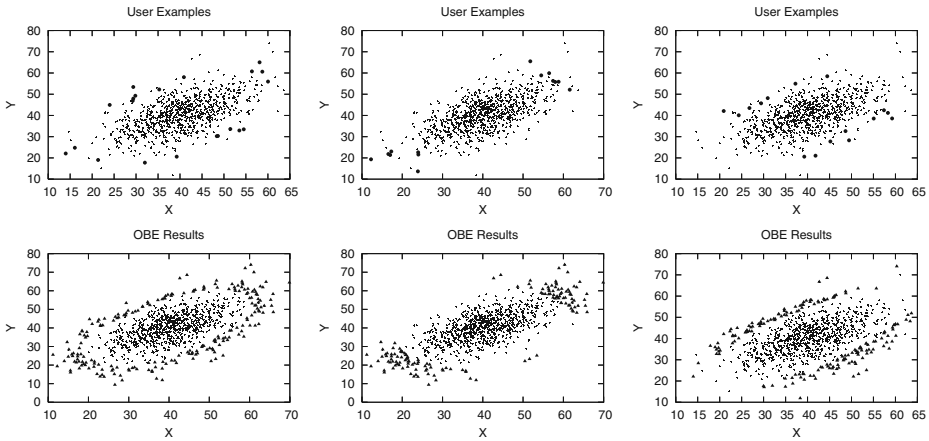
Ellipse dataset The Ellipse dataset has three kinds of interesting outliers as well as outstanding outliers: (i) the set of fringe outliers whose MDEF values are examined across a wide range from 5 to 30, (ii) those mainly spread at the long ends of the ellipse and that display outlierness in two ranges of scales (from 15 to 25 and from 30 to 40), and (iii) mainly in the short ends, which are exceptional in the range from 5 to 15, but do not show strong outlierness in the scales from 35 to 40.

Figure 7a shows the performance of the OBE-RF scheme. From top to bottom, we show the user examples and the detected results for cases E-F, E-L and E-S (see Table 2 for a description of the cases). Note that the features chosen can capture the notion of fringe, long ends, and short ends. Beyond that, OBE can almost perfectly reconstruct these hidden outlier notions.

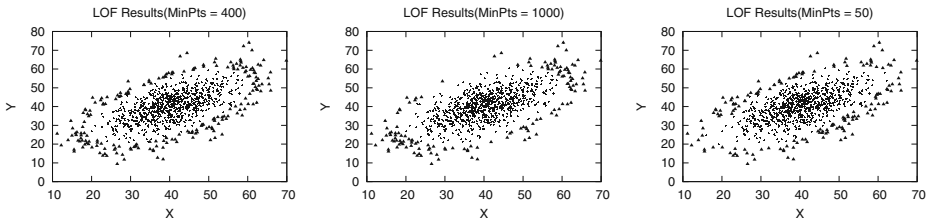
The detailed convergence of the detected fractions and corresponding F1 measurements during iterations of OBE-Fraction are shown in Fig. 8. In Fig. 8, the detected fraction of outliers decreases and converges rapidly to that specified by the user in all three cases. The F1 measurement improves rapidly together with the convergence of the fraction. Note that only four iterations are needed to reach convergence in the E-L experiment.

For comparison, the corresponding detection results of LOF in cases E-F, E-L, and E-S are displayed from left to right in Fig. 7b. Note that a large range of choices for the parameter *MinPts* is tested. All results in Fig. 7b are the best performances of LOF (i.e., the results with the best F1 measurements) and are obtained by using different values of *MinPts*. It is obvious that LOF has poor performance when discovering objects isolated in the short ends of the Ellipse dataset.

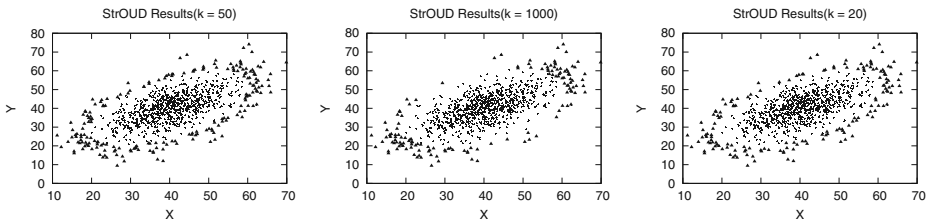
The best experimental results of StrOUD applied to the Ellipse dataset's three cases are summarized in Fig. 7c. Again, the best results are obtained by using different values of K after searching the large range of choices for K. Although the results of cases E-F and E-L are comparable to those obtained by using OBE and LOF, StrOUD could hardly discriminate outliers that were scattered in the short ends of the Ellipse dataset from those in the long ends.



(a) Detection results of OBE-RF on the ellipse dataset. From left to right: case E-F, case E-L, and case E-S; see Table 2 for the description of each case.



(b) Detection results of LOF on the ellipse dataset. From left to right: obtained when *Min Pts* is 400, 1000, and 50, respectively.



(c) Detection results of StrOUD on the ellipse dataset. From left to right: obtained when K is 50, 1000, and 20, respectively.

Fig. 7 Detection results of OBE-RF, LOF, and StrOUD on the ellipse dataset. From *left* to *right*: case E-F, case E-L, and case E-S; see Table 2 for the description of each case. Note that results of LOF and StrOUD are the best after searching a large range of choices for parameters of *Min Pts* in LOF and K in StrOUD

Table 3 Results of OBE's two schemes, LOF and StrOUD, on all datasets

Test data		OBE-fraction				OBE-RF				LOF		StrOUD	
		Prec.	Rec.	F1	#I	Prec.	Rec.	F1	#F	F1	MinPts	F1	k
Ellipse dataset	E-F	90.8 ± 2.5	93.8 ± 3.3	92.2 ± 2.6	7.2 ± 1.2	91.8 ± 10.2	97.7 ± 1.6	94.7 ± 6.5	6.6 ± 3.9	98.1	400	91.0	50
	E-L	91.1 ± 2.4	95.9 ± 1.6	93.4 ± 1.0	4.6 ± 1.0	96.3 ± 4.2	98.3 ± 1.7	97.3 ± 2.7	4.7 ± 2.4	78.6	1,000	85.0	1,000
	E-S	69.0 ± 9.5	82.5 ± 8.0	75.1 ± 6.0	14.5 ± 4.3	71.4 ± 7.8	86.7 ± 7.4	78.3 ± 4.8	2.5 ± 1.8	55.6	50	33.3	20
Mixture dataset	M-A	83.5 ± 3.9	90.9 ± 3.7	87.0 ± 3.0	4.7 ± 0.6	85.9 ± 9.7	92.7 ± 4.8	89.2 ± 5.6	5.4 ± 4.4	93.3	400	72.2	40
	M-L	91.6 ± 3.8	95.5 ± 1.6	93.5 ± 2.4	5.1 ± 1.6	92.8 ± 4.9	97.1 ± 3.2	94.9 ± 3.1	3.9 ± 2.7	90.6	1,000	97.4	100
	M-S	84.6 ± 4.7	89.7 ± 4.2	87.1 ± 2.9	6.1 ± 1.3	78.0 ± 10.1	92.4 ± 5.6	84.6 ± 6.1	2.9 ± 1.8	56.9	60	20.9	2
NYWomen dataset	N-FS	77.4 ± 6.2	83.1 ± 4.1	80.2 ± 4.7	6.6 ± 1.0	87.3 ± 4.9	85.4 ± 8.3	86.3 ± 4.8	3.9 ± 3.2	86.5	700	91.0	400
	N-PF	69.0 ± 5.4	72.8 ± 4.2	70.8 ± 4.7	8 ± 1.4	74.9 ± 3.6	78.1 ± 7.5	76.5 ± 4.4	4.2 ± 3.1	74.0	400	76.7	100
	N-SS	63.1 ± 6.7	71.5 ± 6.6	67.1 ± 5.4	10.1 ± 1.8	62.1 ± 9.9	73.8 ± 7.7	67.5 ± 5.6	2.2 ± 1.2	45.2	20	17.4	2
Abalone dataset	A-FS	65.4 ± 7.6	76.5 ± 6.8	70.5 ± 6.5	5.1 ± 0.8	84.3 ± 8.9	75.2 ± 11.0	79.5 ± 9.7	5.5 ± 4.6	79.4	50	42.2	2
	A-HA	51.5 ± 3.9	61.5 ± 7.0	56.0 ± 4.5	7.1 ± 1.3	71.7 ± 12.3	67.1 ± 13.1	69.3 ± 8.6	2.7 ± 1.4	72.1	700	83.0	60

Precision (Prec.), recall (Rec.), and F1 show the performance of OBE's two schemes. The number of iterations (#I) in OBE-fraction and feedbacks (#F) in OBE-RF for convergence in the detection procedure are also shown

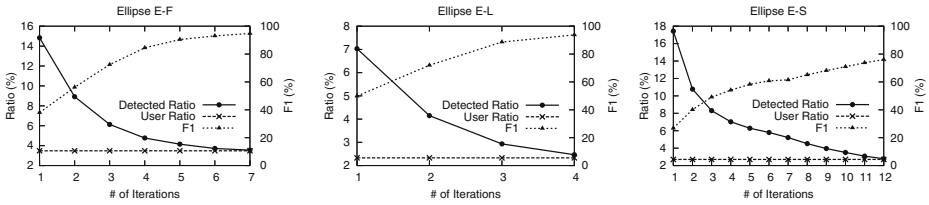


Fig. 8 Convergence of outlier fractions and F1 measurements of OBE-fraction on the ellipse dataset. From *left* to *right*: case E-F, case E-L, and case E-S; see Table 2 for the description of each case

In Fig. 7, both StrOUD and LOF can not highlight those outliers that are isolated at the short ends of the Ellipse dataset and that appear to be meaningful, whereas OBE can.

Mixture dataset For the mixture dataset, we mimicked three categories of interesting outliers: (i) the set of outliers scattered along the fringes of both clusters, (ii) those mainly spread along the fringe of the large cluster, and (iii) those mainly in the small cluster. The results of the OBE-RF scheme in the three cases are shown in Fig. 9. The details of the convergence of OBE-Fraction are in Fig. 10.The features chosen can capture several different and interesting types of outlying objects, and OBE again discovers the underlying outlier notion. It also converges rapidly, needing no more than four iterations to reach convergence in the M-A experiment.

NYWomen dataset In the real dataset, we simulate three kinds of intentions for outliers. The first group (case N-FS; see Fig. 11) is the set of consistently fast or slow runners (i.e., the fastest 7 and most of the remaining 70 were very slow ones). The second group of outlying runners (case N-PF; see Fig. 12) are mainly those who are at least partly fast. In this group, we discover both the fastest 23 runners and abnormally fast runners in one or two parts of the four stretches, although they ranked middle

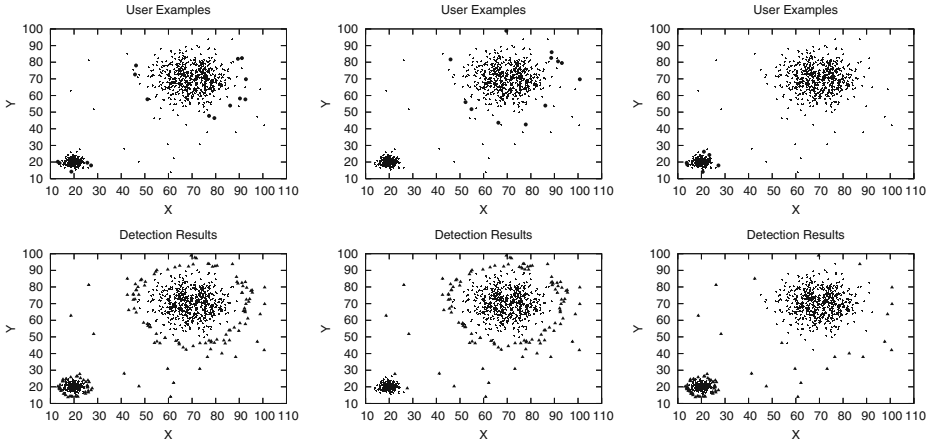


Fig. 9 Detection results of OBE-RF on the mixture dataset. From *left* to *right*: case M-A, case M-L, and case M-S; see Table 2 for the description of each case

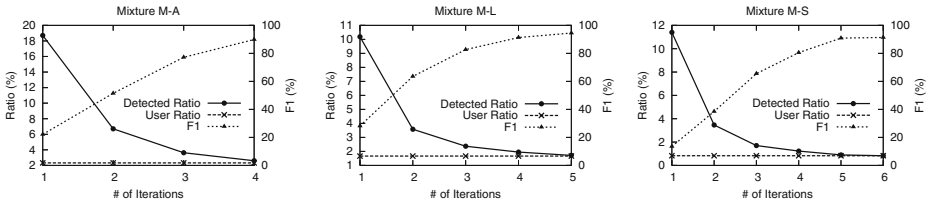


Fig. 10 Convergence of outlier fractions and F1 measurements of OBE-fraction on the mixture dataset. From *left* to *right*: case M-A, case M-L, and case M-S; see Table 2 for the description of each case

or last in the overall race. For example, the runner who ranked 1,275 in the whole race took 47 min for the first 6.2 miles, but took 91 min for the last 6.2 miles. The third set of interesting outliers (case N-SS; see Fig. 13) is detected when we focus on a small scale. Although they ranked middle in the whole race, they exhibit great outlierness compared with their local neighbors. They ran quickly when most of their neighbors ran slowly, and vice versa. Note that in Figs. 11–13 the filled-dot objects

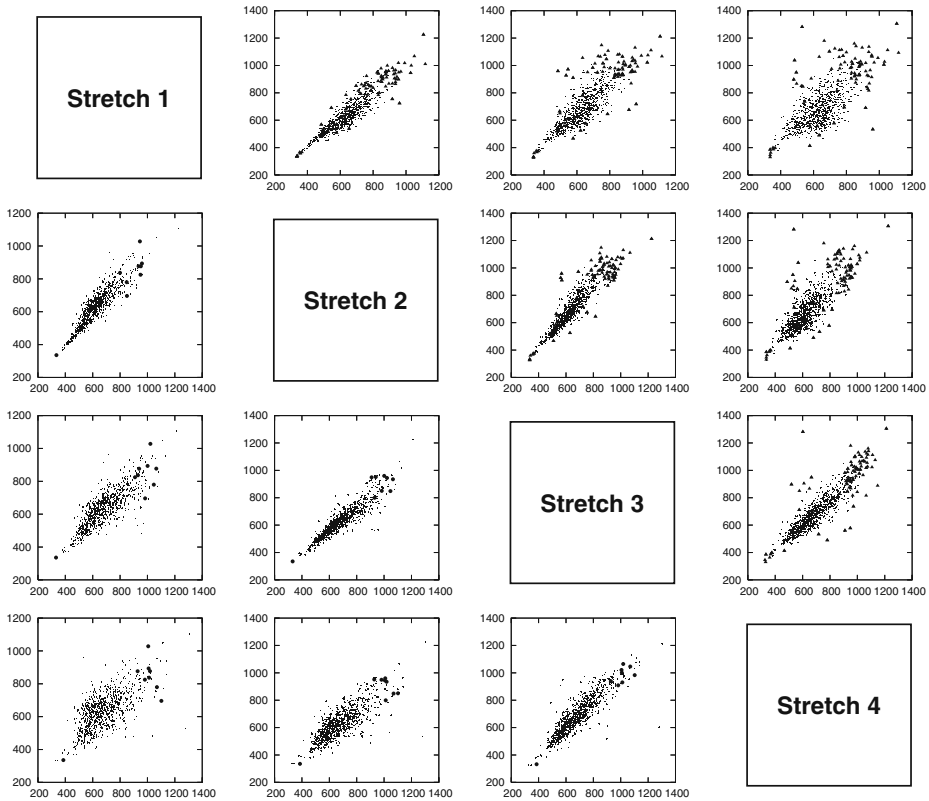


Fig. 11 User examples (*dots*) and outliers (*triangles*) detected by OBE-RF in case N-FS of the NYWomen dataset

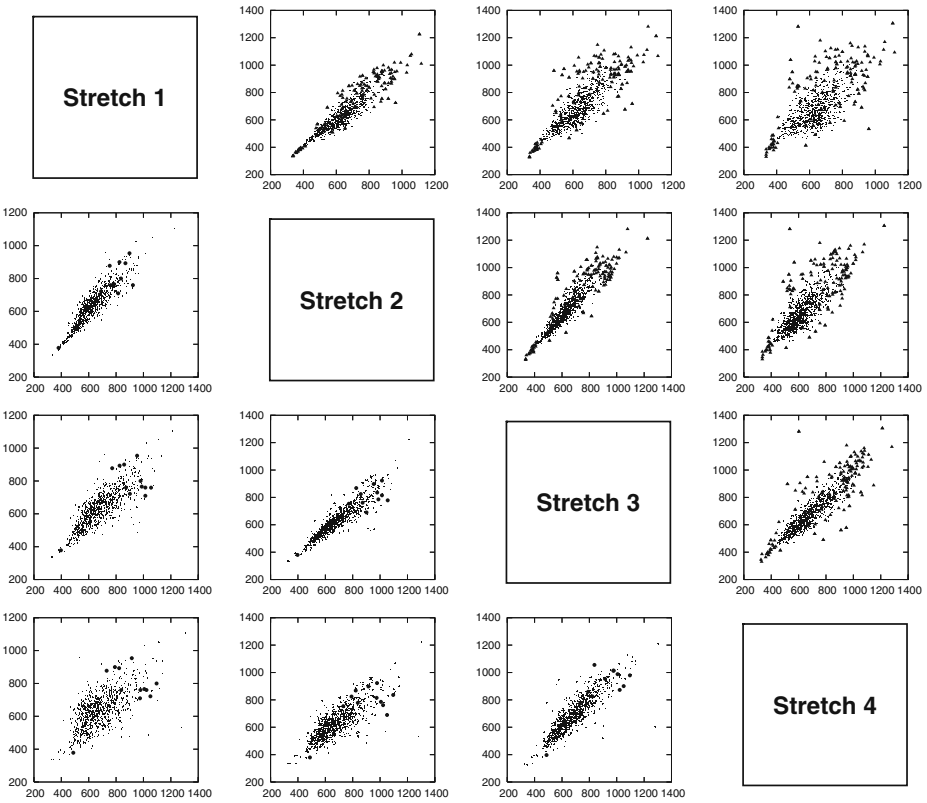


Fig. 12 User examples (*dots*) and outliers (*triangles*) detected by OBE-RF in case N-PF of the NYWomen dataset

are user examples while the outliers are represented by filled triangles. Descriptions of the three cases are given in Table 2. The convergence details are shown in Fig. 14.

Abalone dataset In the abalone dataset obtained from the UCI (http://www.ics.uci.edu/~mlearn/MLRepository.html) machine learning repository, we detect outliers from three attributes: diameter, whole weight, and rings. The outliers were categorized into two quite different groups. The first group (case A-FS; see Fig. 15) is the set of abalones (containing about 90%) that are very fat or very slim. Members of the second set of outliers have an abnormal relationship between their size (diameter and whole weight) and rings (case A-HA; see Fig. 16). The number of rings indicates age. In the second group, some abalones are extremely old compared with the like-sized majority. Others, though young, are huge, displaying very large diameter and weight. Figures 15 and 16 show the two sets of outliers in three dimensions. Furthermore, in Figs. 15 and 16 the filled-dot objects are user examples and the detected outliers are illustrated by using filled triangles. The two cases are described in Table 2. Figure 17 shows the convergence processes.

For all datasets, the results are summarized in Table 3. Table 3 shows the precision, recall, and F1 measurements, including their standard deviations, for both OBE-

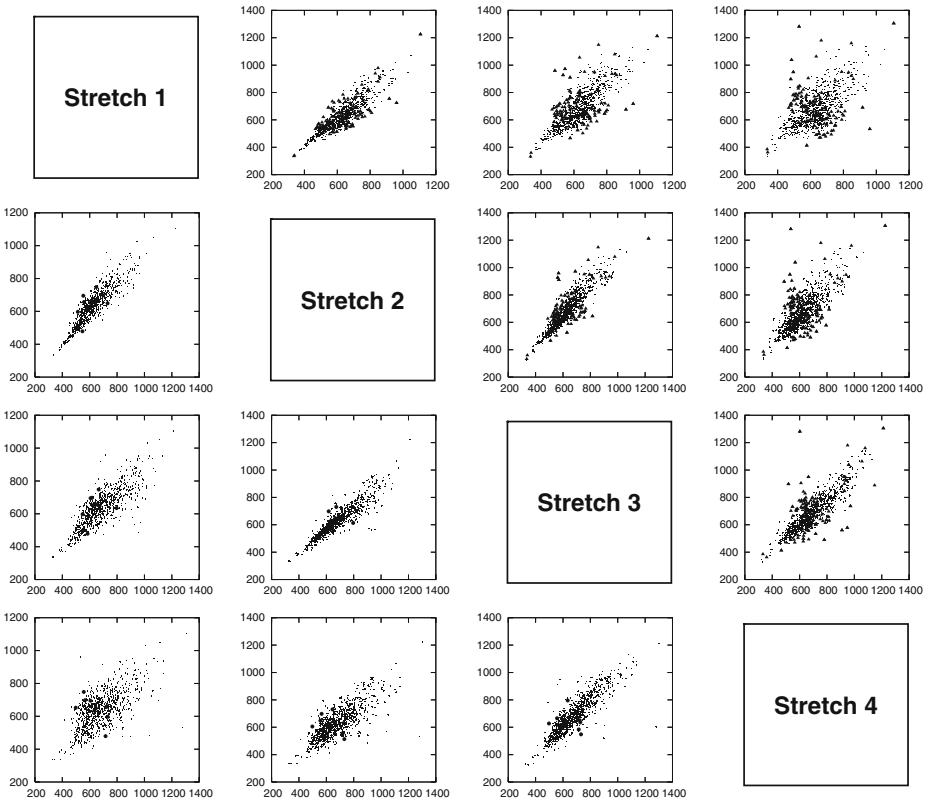


Fig. 13 User examples (*dots*) and outliers (*triangles*) detected by OBE-RF in case N-SS of the NYWomen dataset

Fraction and OBE-RF, by using polynomial kernels (as mentioned, polynomial kernels consistently performed better in our experiments). It also shows the number of iterations and feedbacks needed to converge in the learning step of OBE. In Table 3, all measurements of OBE-Fraction and OBE-RF are the averages of ten trials with different sets of user examples. In each trial, the same examples are used for fair comparisons of the performances of the OBE-Fraction and OBE-RF

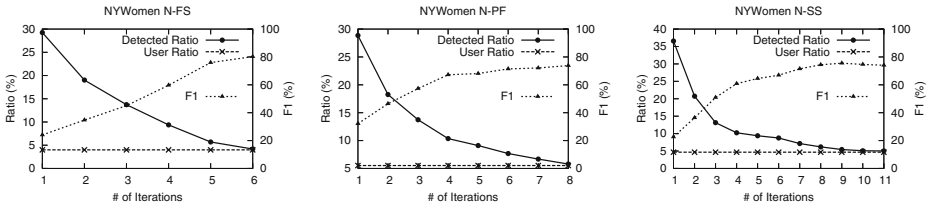


Fig. 14 Convergence of outlier fractions and F1 measurements of OBE-Fraction on the NYWomen dataset. From *left* to *right*: case N-FS, case N-PF, and case N-SS; see Table 2 for the description of each case

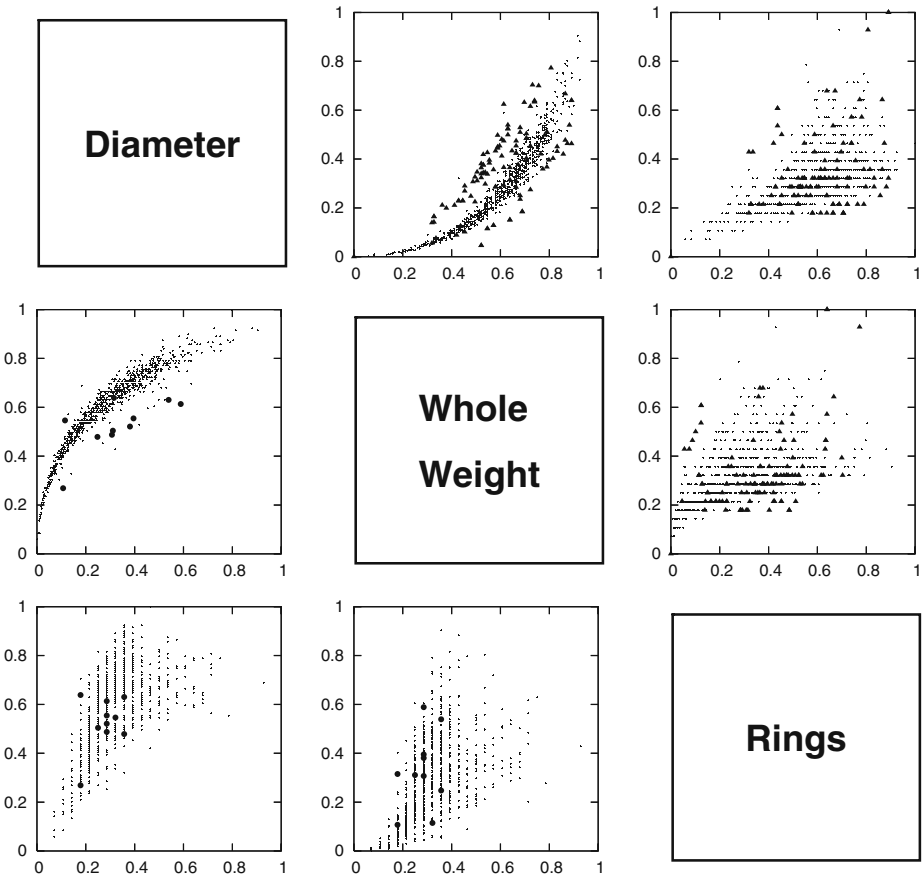


Fig. 15 User examples (*dots*) and outliers (*triangles*) detected by OBE-RF in case A-FS of the Abalone dataset

schemes. The OBE-Fraction scheme detects outliers by using the outlier fraction parameter. The OBE-RF scheme detects outliers by using relevance feedback but does not require the fraction factor.

For comparison, Table 3 also shows the best results (i.e., the results with the best F1 measurements) of running the LOF and StrOUD techniques on all datasets after examining a large range of choices for the parameters *MinPts* in LOF and K in StrOUD. Since the precision, recall, and F1 measurements of LOF/StrOUD are almost equal, only the F1 measurements and the corresponding parameter values are listed in Table 3.

By comparing the performances of the OBE-Fraction and OBE-RF schemes, we observe that OBE-RF tends to give better performance in almost all cases except M-S, even without the information on outlier fractions. This happens because, by using a relevance feedback technique that enables the user to refine preferences by specifying relevant and non-relevant outliers, the hyperplane in the OBE-RF scheme is fitted and set as close as possible to the set of user-provided examples, including

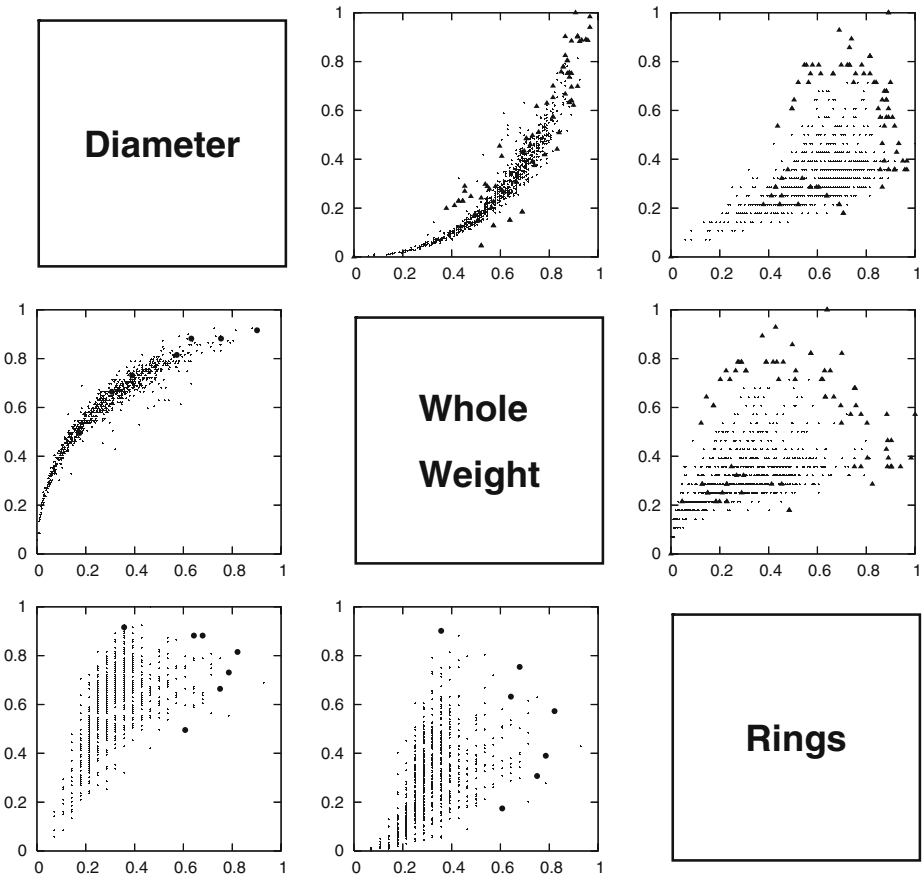


Fig. 16 User examples (*dots*) and outliers (*triangles*) detected by OBE-RF in case A-HA of the Abalone dataset

positive feedbacks. On the other hand, standard deviations of precision, recall, and F1 measurements of OBE-Fraction are usually (26 of 33 measurements) less than those of OBE-RF, indicating that OBE-Fraction is more stable than OBE-RF, with information of the outlier fraction.

As for the comparison of OBE with LOF and StrOUD, OBE-RF is the best for five of the eleven cases. Furthermore, in three cases OBE-RF performs much better than LOF and StrOUD (e.g., for the case E-S, F1 measurements of OBE-RF, LOF and StrOUD are: 79.3, 55.6, and 33.3%; for the cases M-S and N-SS, the results are: 84.6, 56.9, 20.9, 67.5, 45.2, and 17.4%). The results of OBE-RF rank second and are comparable to the best of the three methods in four other cases. In only two cases was OBE-RF the worst of those studied. However, the difference is within a reasonable range (for the cases N-FS and A-HA, the F1 measurements of OBE-RF, LOF and StrOUD are: 86.3, 86.5, 91.0, 69.3, 72.1, and 83.0%).

In almost all cases, OBE's two schemes detect interesting outliers with precision, recall, and F1 measurements reaching from 60 to 90%. In the worst case (case

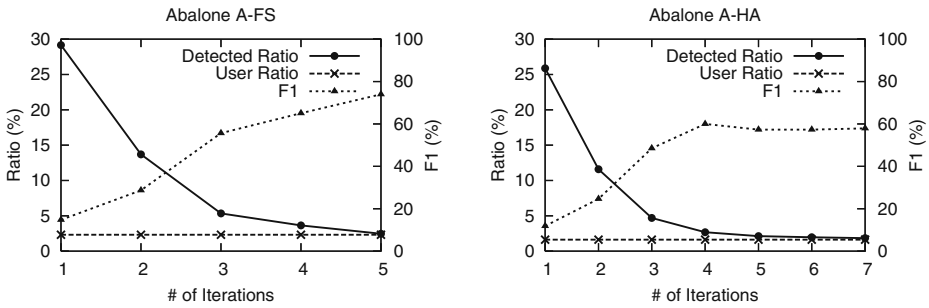


Fig. 17 Convergence of outlier fractions and F1 measurements of OBE-fraction on the Abalone dataset. *Left case* A-FS, *right case* A-HA; see Table 2 for the description of each case

A-HA of Abalone) using the OBE-Fraction scheme, it still achieves 52% precision, 62% recall, and 56% F1. The number of iterations in the OBE-Fraction scheme is always small (less than 15), as is the number of feedbacks (always less than seven in the OBE-RF scheme). As for the standard deviations of all measurements of both schemes, 64% of them are less than 5% and 94% of them are less than 10%. These experiments demonstrate that OBE works well in detecting outliers in the MDEF feature space, even though the number of user examples is very small.

6 Observations

This section presents several experimental observations related to the application of OBE. Specifically, we always use the same sets of user-provided examples and all results are an average of ten trials.

1. *Possible use of non-SVM learning method in OBE*
   Several supervised learning methods, such as naive Bayes and Fisher's linear discriminant analysis (LDA), are based on probabilities within the training data. However, in the context of outlier detection there are only few positive training data (i.e., outlier examples). It is impossible to estimate the appropriate probability from such a small sample. We attempted to use LDA to classify outliers and normal objects. However, this attempt failed because outliers are obviously not normally distributed and the pooled sample covariance matrix of the two classes did not exist.
   Other methods (e.g., decision trees, neural networks, and perception) do not have the maximizing margin feature. As discussed in Section 4.6, SVM maximizes the margin between the positive and negative training data. Under the assumption that SVM does not misclassify separable positive (i.e., outlier) training data,[10] the final hyperplane converges to the true position, where interesting

[10]This is the fact in all our experiments.

outliers and normal objects are separated on either side. Other methods do not guarantee convergence of the separating hyperplane.

2. *Choice of kernel functions*
SVMs provide both linear and nonlinear transformation of input space to feature space by using various kernels. Aside from the polynomial kernel, linear- and radial-based kernels are also frequently chosen. We extensively compare the accuracy and stability of linear, polynomial, and radial basis SVM kernels. Figure 18 shows the performance of the OBE-Fraction scheme by using various kernels of SVM.
The figure reveals several points. (1) Linear kernels sometimes become very poor in their ability to detect outliers because classifying interesting outliers in a high-dimensional MDEF-based space is often not linearly separable. (2) Radial kernels with rigorously chosen gamma parameters (denoted as *gm* in Fig. 18) can achieve better generalization. However, the gamma parameters must be tuned carefully for problem domains and even for cases within a dataset. (3) Polynomial kernels always perform relatively well and are stable in all cases of our experiments. The performance of polynomial kernels without carefully chosen parameters is significant.
Therefore, we used polynomial kernels with the same SVM parameters in all of the experiments in Section 5. Correspondingly, all processes were done automatically. For some particular application domains, however, users might employ the radial kernels in SVMs and carefully choose the gamma parameters to get better performance of OBE. Once the best kernel and parameter are

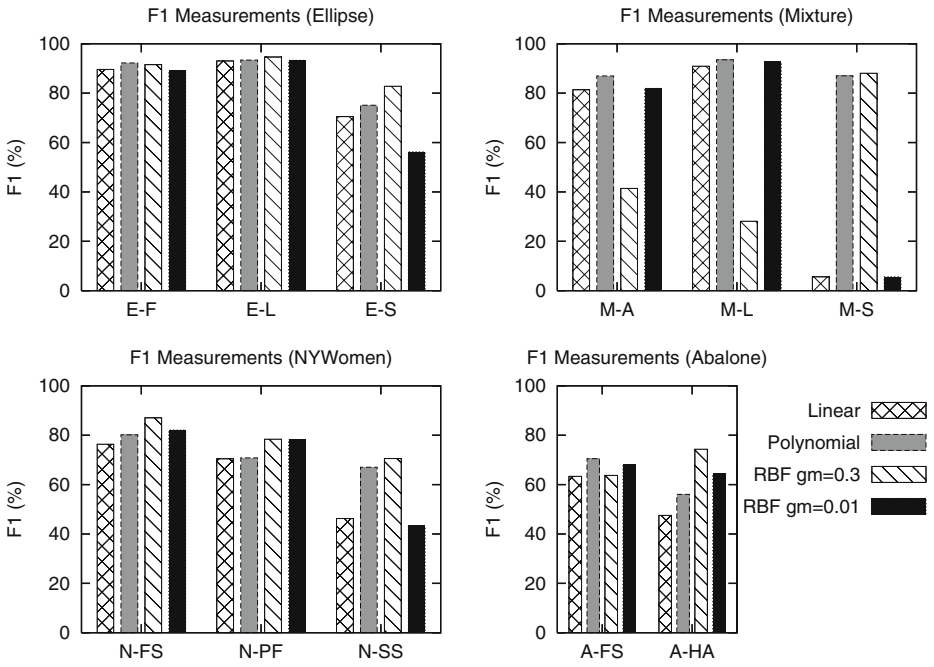


Fig. 18 F1 measurements versus kernels of SVM in OBE-fraction

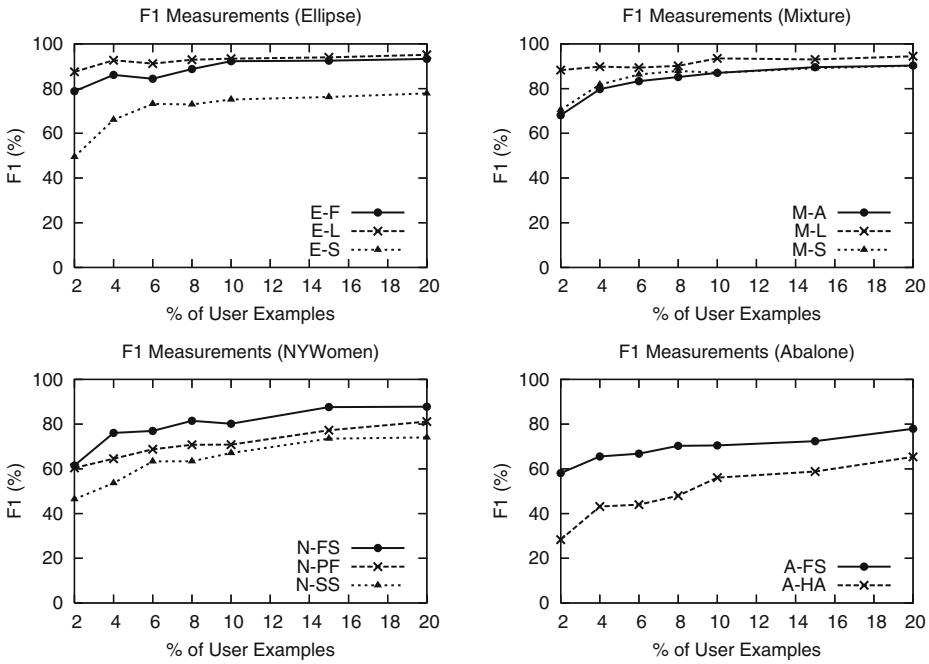


Fig. 19 F1 measurements versus % of user examples in OBE-fraction

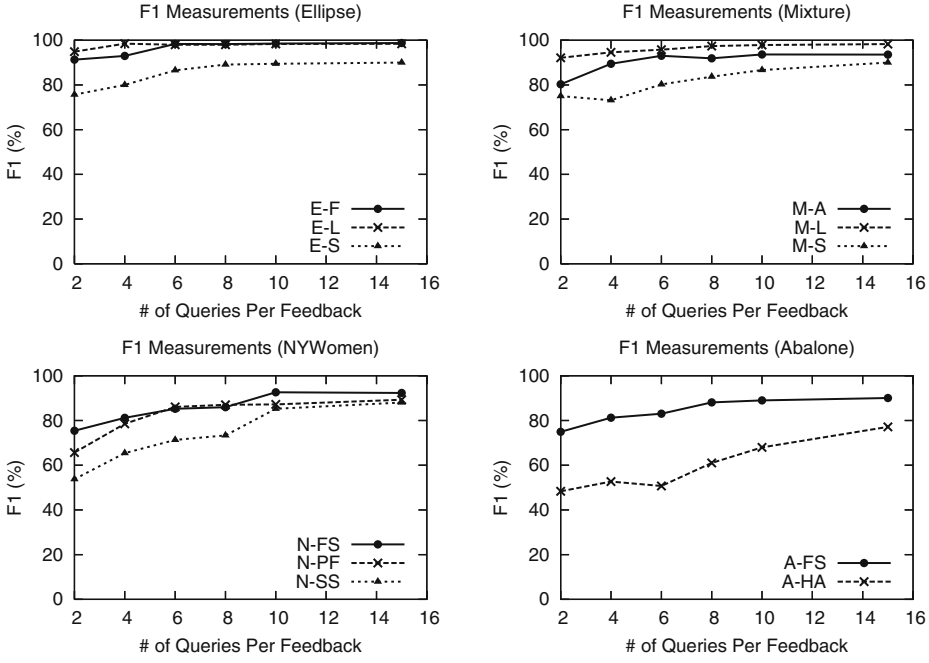


Fig. 20 F1 measurements versus # of queries raised in each feedback sub-step in OBE-RF

determined, OBE can use them automatically to detect outliers for the similar application datasets.

3. *Impact of user examples*
   In the experiments in Section 5, we always sample 10% of the interesting outliers to serve as user-provided examples and hide the rest. More experiments were conducted that sampled examples from 2% to 20% ($y = 2 - 20\%$) to determine the influence of the number of user examples on OBE performance. The results in Fig. 19 are obtained by using the OBE-Fraction scheme. They show that F1 measurements benefit from more examples, as one would expect, then tend to grow more slowly with additional examples. Note that, in some cases such as the E-L case in the Ellipse dataset, OBE performs quite well even with few examples.
4. *Effect of the number of queries per feedback in OBE-RF*
   It is interesting to ask whether more queries in each feedback sub-step lead to better performance of the OBE-RF scheme. We added experiments to answer this question. Figure 20 shows the performance of the OBE-RF scheme when 2–15 queries ($m = 2 - 15$) are raised in each feedback sub-step. The results in Fig. 20 are an average of ten trials when different user examples of 4%($y = 4\%$) are sampled. The results show that the OBE-RF scheme benefits from more queries per feedback as well.

7 Conclusion

Detecting outliers is an important but tricky problem since the exact definition of an outlier often depends on the user and/or the dataset. We proposed to solve this problem by using a novel approach: that of having the user provide examples of outliers.

This paper contributes the following:

- We proposed OBE, which, to the best of our knowledge, is the first method to provide a solution to this problem.
- We built a system and described our design decisions. Although OBE appears simple to the user (click on a few outlier-looking record), there are many technical challenges that needed to be resolved. We showed how to approach them and, specifically, how to extract suitable feature vectors from our data objects. We also illustrated how to train a classifier quickly to learn from the (few) examples that the user provides. Two schemes, OBE-Fraction and OBE-RF, were developed especially for different degrees of domain knowledge.
- We evaluated the two OBE schemes by using both real and synthetic data, with several small sets of user examples. Our experiments demonstrated that both the OBE-Fraction and OBE-RF schemes can successfully incorporate these examples in the discovery process and detect outlierness characteristics very similar to the given examples.

Acknowledgements This research was supported in part by the Japan-U.S. Cooperative Science Program of JSPS, U.S.-Japan Joint Seminar (NSFgrant0318547), the Grant-in-Aid for Scientific Research from JSPS (#15300027), and the Beijing outstanding talents training and subsidization.

References

Aggarwal, C. C., & Yu, P. S. (2001). Outlier detection for high dimensional data. In *Proc. SIGMOD*.

Angiulli, F., & Pizzuti, C. (2005). Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 203–215.

Barbará, D., Domeniconi, C., & Rogers, J. P. (2006). Detecting outliers using transduction and statistical testing. In *Proc. SIGKDD conf.* (pp. 55–64).

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data*. New York: Wiley.

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is nearest neighbors meaningful? In *Proc. international conf. on database theory* (pp. 217–235).

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proc. SIGMOD Conf.* (pp. 93–104).

Goh, K., Chang, E., & Cheng, K. (2001). SVM binary classifier ensembles for image classification. In *Proc. International conf. on information and knowledge management* (pp. 395–402).

Hawkins, D. M. (1980). *Identification of outliers*. London, UK: Chapman and Hall.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264–323.

Joachims, T. (1998). Text categorization with support vector machines. In *Proc. European conf. machine learning (ECML)* (pp. 137–142).

Johnson, T., Kwok, I., & Ng, R. T. (1998). Fast computation of 2-dimensional depth contours. In *Proc. KDD* (pp. 224–228).

Knorr, E. M., & Ng, R. T. (1997). A unified notion of outliers: Properties and computation. In *Proc. KDD* (pp. 219–222).

Knorr, E. M., & Ng, R. T. (1998). Algorithms for mining distance-based outliers in large datasets. In *Proc. VLDB* (pp. 392–403).

Knorr, E. M., & Ng, R. T. (1999). Finding intentional knowledge of distance-based outliers. In *Proc. VLDB* (pp. 211–222).

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: Algorithms and applications. *VLDB Journal, 8*(3–4), 237–253.

Markowetz, F. (2003). *Support vector machines in bioinformatics*. Ph.D. Thesis, University of Heidelberg.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., & Faloutsos, C. (2003). LOCI: Fast outlier detection using the local correlation integral. In *Proc. ICDE* (pp. 315–326).

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *Proc. ACM SIGMOD international conference on management of data* (pp. 427–438).

Rousseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. New York: Wiley.

Tax, D. M. J., & Duin, R. P. W. (1999). Support vector domain description. *Pattern Recognition Letters, 20*, 1991–1999.

Yamanishi, K., & Takeuchi, J. (2001). Discovering outlier filtering rules from unlabeled data. In *Proc. KDD* (pp. 389–394).

Yamanishi, K., Takeuchi, J., Williams, G., & Milne, P. (2000). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proc. KDD* (pp. 250–254).

Yu, H., Han, J., & Chang, K. (2002). PEBL: Positive example based learning for web page classification using SVM. In *Proc. KDD* (pp. 239–248).

Zhu, C., Kitagawa, H., & Faloutsos, C. (2005). Example-based robust outlier detection in high dimensional datasets. In *Proc. ICDM* (pp. 829–832).

Zhu, C., Kitagawa, H., Papadimitriou, S., & Faloutsos, C. (2004). OBE: Outlier by example. In *Proc. PAKDD* (pp. 222–234).