OBE: Outlier by Example

Cui Zhu[1], Hiroyuki Kitagawa[2], Spiros Papadimitriou[3], and Christos Faloutsos[3]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba
[2] Institute of Information Sciences and Electronics, University of Tsukuba
{zhucui, kitagawa}@kde.is.tsukuba.ac.jp
[3] School of Computer Science, Carnegie Mellon University
{spapadim+, christos}@cs.cmu.edu

Abstract. Outlier detection in large datasets is an important problem. There are several recent approaches that employ very reasonable definitions of an outlier. However, a fundamental issue is that the notion of which objects are outliers typically varies between users or, even, datasets. In this paper, we present a novel solution to this problem, by bringing users into the loop. Our *OBE (Outlier By Example)* system is, to the best of our knowledge, the first that allows users to give some examples of what they consider as outliers. Then, it can directly incorporate a *small* number of such examples to successfully discover the hidden concept and spot further objects that exhibit the same "outlier-ness" as the examples. We describe the key design decisions and algorithms in building such a system and demonstrate on both real and synthetic datasets that OBE can indeed discover outliers that match the users' intentions.

1 Introduction

In many applications (e.g., fraud detection, financial analysis and health monitoring), rare events and exceptions among large collections of objects are often more interesting than the common cases. Consequently, there is increasing attention on methods for discovering such "exceptional" objects in large datasets and several approaches have been proposed.

However, the notion of what is an *outlier* (or, exceptional/abnormal object) varies among users, problem domains and even datasets (problem instances): (i) different users may have different ideas of what constitutes an outlier, (ii) the same user may want to view a dataset from different "viewpoints" and, (iii) different datasets do not conform to specific, hard "rules" (if any).

We consider objects that can be represented as multi-dimensional, numerical tuples. Such datasets are prevalent in several applications. From a general perspective [4,7,8,2], an object is, intuitively, an *outlier* 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 provide different sets of outliers.

Typically, users are experts in their problem domain, not in outlier detection. However, they often have a few example outliers in hand, which may "describe"

H. Dai, R. Srikant, and C. Zhang (Eds.): PAKDD 2004, LNAI 3056, pp. 222–234, 2004.

their intentions and they want to find more objects that exhibit "outlier-ness" characteristics similar to those examples. Existing systems do not provide a direct way to incorporate such examples in the discovery process.

Example. We give a concrete example to help clarify the problem. The example is on a 2-d vector space which is easy to visualize, but ideally our method should work on arbitrary dimensionality or, even, metric datasets[1].

Consider the dataset in Figure 1. In this dataset, there are a large sparse cluster, a small dense cluster and some clearly isolated objects. Only the isolated objects (circle dots) are outliers from a "bird's eye" view. In other words, when we examine wide-scale neighborhoods (i.e., with large radius—e.g., covering most of the dataset), only the isolated objects have very low neighbor densities, compared with objects in either the large or the small cluster. However, consider the objects on the fringe of the large cluster (diamond dots). These can also be regarded as outliers, if we look closer at mid-scale (i.e., radius) neighborhoods. Also, objects on the fringe of the small cluster (cross dots) become outliers, if we further focus into small-scale neighborhoods. As exemplified here, different objects may be regarded as outliers, depending on neighborhood scale (or, size).

This scenario is intuitive from the users' perspective. However, to the best of our knowledge, none of the existing methods can directly incorporate user examples in the discovery process to find out the "hidden" outlier concept that users may have in mind.

In this paper, we propose *Outlier By Example (OBE)*, an outlier detection method that can do precisely that: discover the desired "outlier-ness" at the appropriate scales, based on a small number of examples. There are several challenges in making this approach practical; we briefly list the most important: **(1)** What are the appropriate features that can capture "outlier-ness?" These should ideally capture the important characteristics *concisely* and be efficient to compute. However, feature selection is only the tip of the iceberg. **(2)** Furthermore, we have to carefully choose exactly what features to extract. **(3)** The method should clearly require minimal user input and effectively use a *small* number of positive examples in order to be practical. Furthermore, it should ideally not need negative examples. **(4)** Given these requirements, can we train a classifier using only the handful of positive examples and unlabeled data? In the paper we describe the key algorithmic challenges and design decisions in detail.

In summary, the main contributions of this paper are: **(1)** We introduce example-based outlier detection. **(2)** We demonstrate its intuitiveness and feasibility. **(3)** We propose OBE, which, to the best of our knowledge, is the first method to provide a solution to this problem. **(4)** We evaluate OBE on both real and synthetic data, with several small sets of user examples. Our experiments demonstrate that OBE can succesfully incorporate these examples in the discov-

[1] A metric dataset consists of objects for which we only know the pairwise distances (or, "similarity"), without any further assumptions.

ery process and detect outliers with "outlier-ness" characteristics very similar to the given examples.

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

2 Related Work

In essence, outlier detection techniques traditionally employ unsupervised learning processes. The several existing approaches can be broadly classified into the following categories: *(1) Distribution-based approach.* These are the "classical" methods in statistics [1,11]. *(2) Depth-based approach.* This computes different layers of k-d convex hulls and flags objects in the outer layer as outliers [5]. *(3) Clustering approach.* Many clustering algorithms detect outliers as by-products [6]. *(4) Distance-based approach.* Distance-based outliers [7,8,9,10,3] use a definition based on a single, global criterion. 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,* proposed by M. Breunig, et al. [2]. They introduced a local outlier factor (LOF) for each object, indicating its degree of "outlier-ness." LOF depends on the local density of its neighborhood. The neighborhood is defined by the distance to the MinPts-th nearest neighbor. When we change the value of the parameter MinPts, the degree of "outlier-ness" can be estimated in different scopes. However, LOF is very sensitive to the selection of MinPts values, and it has been proven that LOF cannot cope with the multi-granularity problem. *(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 and flags as outliers those objects whose MDEF values deviate significantly from the local average in neighborhoods of *some* scales. So, even though the definition of MDEF can capture "outlier-ness" in different scales, these differences are up to the user to examine manually.

Another outlier detection method was developed in [15], which focuses on the discovery of rules that characterize outliers, for the purposes of filtering new points in a security monitoring setting. This is a largely orthogonal problem. Outlier scores from SmartSifter are used to create labeled data, which are then used to find the outlier filtering rules.

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

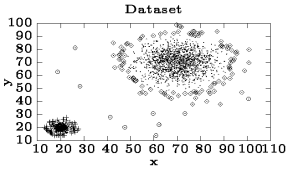

Fig. 1. Illustration of different kinds of outliers in a dataset.

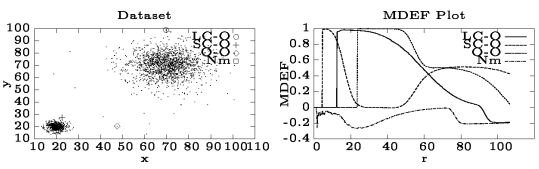

Fig. 2. Illustrative dataset and MDEF plots.

3 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." It is crucial to select features that capture the important characteristics concisely. However, feature selection is only the initial step. In OBE, we employ MDEF for this purpose, which measures "outlier-ness" of objects in the neighborhoods of different scales (i.e., radii).

Detailed definition of the multi-granularity deviation factor (MDEF) is given in [12]. 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.[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, 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, $r_{min} \leq r \leq r_{max}$, where r_{max} is the maximum distance of all object pairs in the given dataset and r_{min} is determined based on the number of objects in the r-neighborhood of p_i. In our experiments, for each p_i in the dataset, r_{min} for p_i (denoted by $r_{min,i}$) is the distance to its 20-th nearest neighbor. In other words, we do not examine the MDEF value of an object until the number of objects in its sampling neighborhood reaches 20. This is a reasonable choice which effectively avoids introduction of statistical errors in MDEF estimates in practice.

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

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

- Consider the point in the middle of the large cluster, Nm, (at about $x = 70$, $y = 68$). The MDEF value is low at *all* scales: compared with its neighborhood, whatever the scale is, the local neighborhood density is always similar to the average local density in its sampling neighborhood. So, the object can be always regarded as a normal object in the dataset.
- In contrast, for the other three objects, there exist situations where the MDEFs are very large, some times even approaching 1. This shows that they differ significantly from their neighbors in *some* scales. The greater the MDEF value is, the stronger the degree of "outlier-ness".

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

- The MDEF value of the outlier in the small cluster, SC-O, (at about $x = 22$, $y = 27$), reaches its maximum at radius $r \approx 5$, then it starts to decrease rapidly until it becomes 0 and remains there for a while (in the range of $r \approx 23--45$). Then the MDEF value increases again but only to the degree of 0.6. The change of MDEF values indicates that the object is extremely 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-O, (at about $x = 70$, $y = 98$), exhibits strong "outlier-ness" in the range from $r = 10$ to $r = 30$, then becomes more and more ordinary as we look at a larger scale.
- For the isolated outlier, O-O, (at about $x = 47$, $y = 20$), its MDEF value stays at 0 up to almost $r = 22$, indicating that it is an isolated object. Then, it immediately displays a high degree of "outlier-ness."

4 Proposed Method (OBE)

4.1 Overview

OBE detects outliers based on user-provided examples and a user-specified fraction of objects to be detected as outliers in the dataset. OBE performs outlier detection in three stages: feature extraction step, example augmentation step and classification step. Figure 3 shows the overall OBE framework.

4.2 Feature Extraction Step

The purpose of this step is to map all objects into the MDEF-based feature space, where the MDEF plots of objects capturing the degree of "outlier-ness," as well as the scales at which the "outlier-ness" appears, are represented by

vectors. Let D be the set of objects in the feature space. In this space, each object is represented by a vector: $O_i = (m_{i0}, m_{i1}, \ldots, m_{in})$, $O_i \in D$, where $m_{ij} = MDEF(p_i, r_j, \alpha r)$, $0 \leq j \leq n$, $r_0 = min_k(r_{min,k})$, $r_n = r_{max}$, $r_j = \frac{r_n - r_0}{n} j + r_0$.[3]

4.3 Example Augmentation Step

In the context of outlier detection, outliers are usually few, and the number of examples that users could offer is even less. If we only learn from the given examples, the information is very little to be used to construct an accurate classifier. However, example-based outlier detection is practical only if the number of required examples is small. OBE effectively solves this problem by augmenting the user-provided examples.

In particular, the examples are augmented by adding outstanding outliers and artificial positive examples, based on the original examples.

Outstanding Outliers. After all objects are projected into the feature space, we can detect outstanding outliers. The set of outstanding outliers is defined by $\{O_i \,|\, max\_M(O_i) > K,\ O_i \in D\}$, where $max\_M(O_i) = max_j(m_{ij})$ and K is a threshold.

Artificial Examples. The examples are further augmented by creating "artificial" data. This is inspired by the fact that an object is sure to be an outlier if *all* of its feature values (i.e., MDEF values) are greater than those of the given outlier examples. Figure 4 shows the created artificial data and the original example.

Artificial data are generated in the following way: (1) Take the difference between the $max\_M(O_i)$ and the threshold K, $Diff\_M(i) = K - max\_M(O_i)$. (2) Divide the difference, $Diff\_M(i)$, into x intervals, where x is the number of artificial examples generated from an original outlier example plus 1. For instance, if the intended augmentation ratio is 200%, two artificial examples are generated from each original example. Then we divide $Diff\_M(i)$ into 3 intervals ($x = 3$), $Intv\_M(i) = Diff\_M(i)/x$. (3) Then, create artificial examples as: $O\_A(i, j)$=$(m_{i0} + j * Intv\_M(i), m_{i1} + j * Intv\_M(i), \ldots, m_{in} + j * Intv\_M(i))$ for $1 \leq j \leq x - 1$. Here, $O\_A(i, j)$ is the j-th artificial example generated from object O_i.

In this way, the "outlier-ness strength" of the user's examples is amplified, in a way consistent with these examples.

Putting together the original examples, outstanding outliers and artificial examples, we get the positive training data.

4.4 Classification Step

So far, the (augmented) positive examples, as well as the entire, unlabeled dataset are available to us. The next crucial step is finding an efficient and

[3] More precisely, if $r_j \geq r_{min,i}$, $m_{ij} = MDEF(p_i, r_j, \alpha r)$, otherwise, $m_{ij} = 0$.

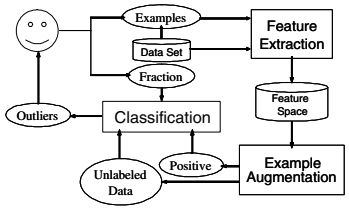

Fig. 3. The Framework of OBE

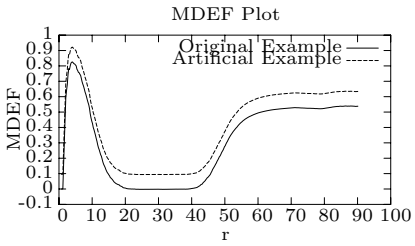

Fig. 4. The Artificial and the Original Examples

effective algorithm to discover the "hidden" outlier concept that the user has in mind.

We use an SVM (Support Vector Machine) classifier to learn the "outlierness" of interest to the user and then detect outliers which match this. Traditional classifier construction needs both positive and negative training data. However, it is too difficult and also a burden for users to provide negative data. Most objects fall in this category and it is unreasonable to expect users to examine them.

However, OBE addresses this problem and can learn only from the positive examples obtained in the augmentation step and the unlabeled data (i.e., the rest of the objects in the dataset). The algorithm shown here uses the marginal property of SVMs. In this sense, it bears some general resemblance to PEBL [13], which was also proposed for learning from positive and unlabeled data. However, in PEBL, the hyperplane for separating positive and negative data is set as close as possible to the set of given positive examples. In the context of OBE, the positive examples are just examples of outliers, and it is not desirable to set the hyperplane as in PEBL. The algorithm here decides the final separating hyperplane based on the fraction of outliers to be detected. Another difference between OBE and PEBL is that strong negative data are determined taking the characteristics of MDEF into consideration.

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

Negative training data extraction sub-step. All objects are sorted in descending order of max_M. Then, from the objects at the bottom of the list, we select a number of (strong) negative training data equal to the number of positive training data. Let the set of strong negative training data be NEG. Also, let the set of positive training data obtained in the example augmentation step be POS.

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

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

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

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

Input:
  Set of outlier examples: E
  Fraction of outliers: F
  Dataset: D
Output:
  Outliers like examples
Algorithm:
  $OO := \emptyset$ *// Outstanding outliers*
  // Feature extraction step:
  For each $p_i \in D$
    For each j $(0 \leq j \leq n)$
      Compute MDEF value m_{ij}
      If $m_{ij} > K$
      Then $OO := OO \cup \{p_i\}$

// Example augmentation step:
For each example in E
  Create artificial examples
$POS := E \cup OO \cup$ artificial examples
// Classification step:
$NEG :=$ strongest negatives
$P := D$
Do {
  $P' := P$
  SVM := train_SVM (POS, NEG)
  (P, N) := SVM.classify (D)
  $NEG := N$
} while ($|P| \geq F * |D|$ and $|P| \neq |P'|$)
return P'

Fig. 5. The Overall Procedure of OBE

Table 1. Description of synthetic and real datasets.

Dataset	Description
Uniform	A 6000-point group following an uniform distribution.
Ellipse	A 6000-point ellipse following a Gaussian distribution.
Mixture	A 5000-point sparse Gaussian cluster, a 2000-point dense Gaussian cluster and 10 randomly scattered outliers.
NYWomen	Marathon runner data, 2229 women from the NYC marathon: average pace (in minutes per mile) for each stretch (6.2, 6.9, 6.9, and 6.2 miles).

Figure 5 summarizes the overall procedure of OBE.

5 Experimental Evaluation

In this section, we describe our experimental methodology and the results obtained by applying OBE to both synthetic and real data, which further illustrate the intuition and also demonstrate the effectiveness of our method.

We use three synthetic and one real datasets (see Table 1 for descriptions) to evaluate OBE.

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 "$<$".

Table 2. Interesting Outliers, Discriminants and the Performance of OBE. OO denotes outstanding outliers, IO denotes interesting outliers. Precision(Preci-), recall(Reca-) and the number of iterations(Iter-) for convergence in the classification step are used to show the performance of OBE.

Dataset	OO	Cases				OBE		
		Label	Discription	Condition	IO	Preci-	Reca-	Iter-
Uniform Dataset	0	U-Fringe	Fringe	(0.3, 0.6, >, 0.4)	330	82.76	88.18	8.1
		U-Corner	Corner	(1, 2, >, 0.5)	274	91.90	97.92	4.1
Ellipse Dataset	15	E-Fringe	Fringe	(5, 30, >, 0.85)	214	90.20	93.55	6.1
		E-Long	Long Ends	(15, 25, >, 0.8) (30, 40, >, 0.6)	140	88.67	92.14	5.4
		E-Short	Short Ends	(5, 15, >, 0.8) (35, 40, <, 0.6)	169	76.46	80.00	10.4
Mixture Dataset	29	M-All	All	(1, 35, >, 0.9)	166	86.32	93.80	4.5
		M-Large	Large Cluster	(15, 35, >, 0.9)	123	91.52	95.37	4.6
		M-Small	Small Cluster	(1, 5, >, 0.9)	72	91.30	97.92	5.3
NYWomen Dataset	17	N-FS	Very Fast/Slow	(800, 1400, >, 0.7)	91	81.53	84.95	6.5
		N-PF	Partly Fast	(300, 500, >, 0.8) (1400, 1600, <,0.4)	126	73.07	78.81	6.9
		N-SS	Stable Speed	(100, 300, >, 0.8) (400, 600, <, 0.3)	121	66.55	70.74	9.2

2. Then, we "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 using OBE.
4. Finally, we compare the detected outliers to the (known) simulated set of outliers. More specifically, we evaluate the success of OBE in recovering the hidden outlier concept using precision/recall measurements.

OBE reports as interesting outliers the outstanding ones, 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 instance, 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. The number of the outstanding outliers and 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.

To detect outstanding outliers, we use $K = 0.97$ for all the synthetic datasets and $K = 0.99$ for the NYWomen dataset. The discovered outstanding outliers of the synthetic datasets are shown in Figure 6. Also, during the augmentation step, we always generate 5 ($x = 6$) artificial examples from each original example.

We use the LIBSVM [14] implementation for our SVM classifier. We extensively compared the accuracy of both linear and polynomial SVM kernels and found that polynomial perform consistently better. Therefore, in all experiments,

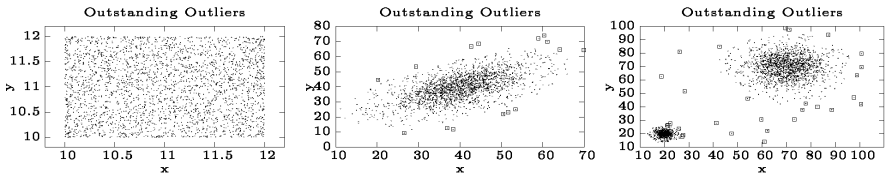

Fig. 6. Outstanding Outliers in the Synthetic Datasets.

we use polynomial kernels and the same SVM parameters[4]. Therefore, the whole processes can be done automatically. We report the effectiveness of OBE in discovering the "hidden" outliers using precision and recall measurements:

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

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

5.2 Results

Uniform dataset. Figure 7 shows the outliers detected by OBE. Although one might argue that no objects from an (infinite!) uniform distribution should be labeled as outliers, the objects at the fringe or corner of the group are clearly "exceptional" in some sense. On the top row, we show the interesting outliers, original examples and the detected results for case U-Fringe. The bottom row shows those for case U-Corner (see Table 2 for a description of the cases). Note that the chosen features can capture the notion of both "edge" and "corner" and, furthermore, OBE can almost perfectly reconstruct these hidden outlier notions!

Ellipse dataset. We simulate three kinds of interesting outliers for the ellipse dataset: (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 OBE is shown in Figure 8. Again, the features can capture several different and interesting types of outlying objects and OBE again discovers the underlying outlier notion!

Mixture dataset. We also mimic three categories of interesting outliers: (i) the set of outliers scattered along the fringe of both clusters, (ii) those maily spread along the fringe of the large cluster, and (iii) those mainly in the small cluster. Due to space constraints, the figure is omitted here.

[4] 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 $(u' * v + 1)^2$.

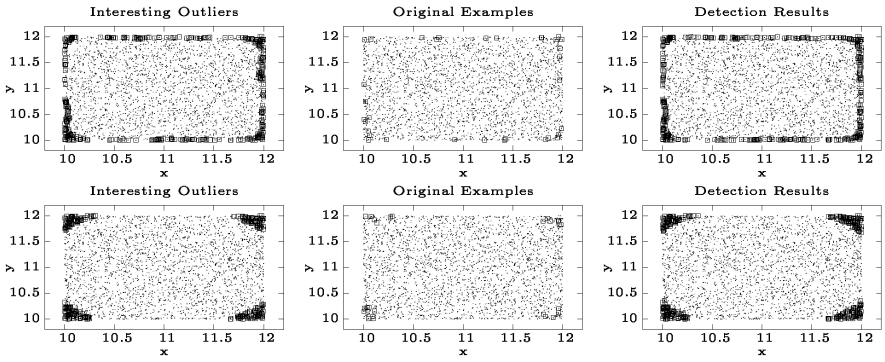

Fig. 7. Detection Results on the Uniform Dataset. Top row: case U-Fringe, bottom row: case U-Corner—see Table 2 for description of each case.

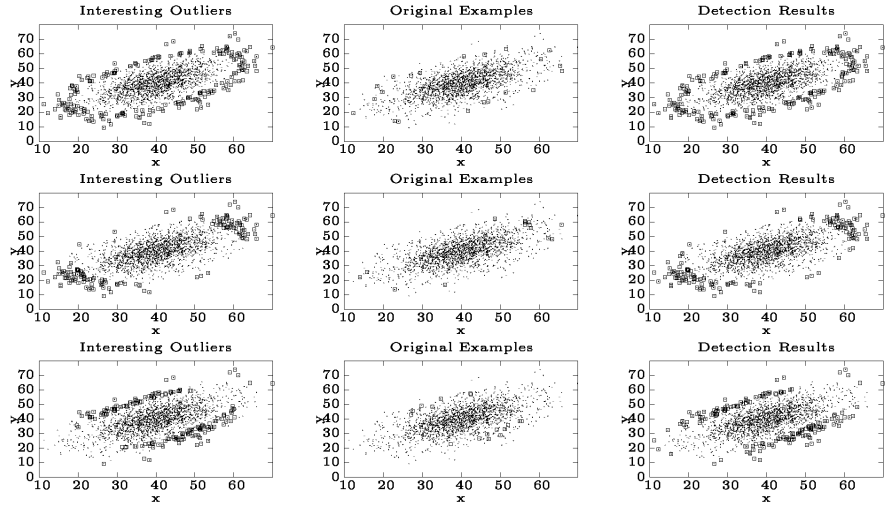

Fig. 8. Detection Results on the Ellipse dataset. From top to bottom, in turn: case E-Fringe, case E-Long, case E-Short—see Table 2 for description of each case.

NYWomen dataset. In the real dataset, we mimic three kinds of intentions for outliers: The first group (case N-FS) is the set of consistently fast or slow runners (i.e., the fastest 7 and almost all of the 70 very slow ones). The second group of outlying runners (case N-PF) are those who are at least partly fast. In this group, we discover both the fastest 23 runners and those runners who were abnormally fast in one or two parts of the four stretches, although they rank middle or last in the whole race. For example, one of them took 47 minutes for the first 6.2 miles, while 91 minutes for the last 6.2 miles. The third set of interesting outliers (case N-SS) is those who run with almost constant speed and rank middle in the whole race. They are very difficult to perceive, but they

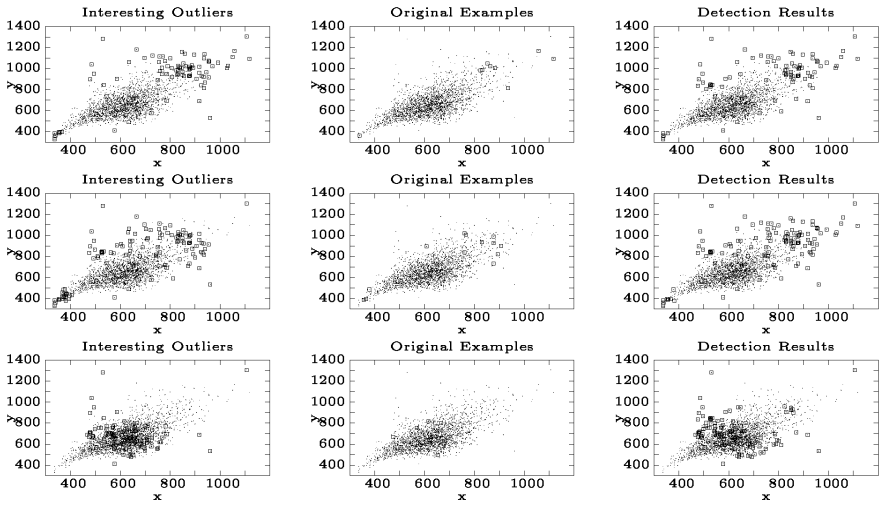

Fig. 9. Detection Results on the NYWomen Dataset. From top to bottom in turn: Case N-FS, Case N-PF, Case N-SS—see Table 2 for description of each case. Only the first and forth dimensions are used for the plots, although NYWomen Dataset is four dimensional.

certainly exhibit "outlier-ness" when we examine them at a small scale. Because of space limits, we only show the result plots in the first and forth dimensions —see Figure 9.

For all datasets, Table 2 shows the precision and recall measurements for OBE, using polynomial kernels (as mentioned, polynomial kernels always performed better than linear kernels in our experiments). It also shows the number of iterations needed to converge in the learning step. In Table 2, all the measurements are averages of ten trials. In almost all cases, OBE detects interesting outliers with both precision and recall reaching 80–90%. In the worst case (case N-SS of NYWomen), it still achieves 66% precision and 70% recall. The number of iterations is always small (less than 10).

6 Conclusion

Detecting outliers is an important, but tricky problem, since the exact notion of an outlier often depends on the user and/or the dataset. We propose to solve this problem with a completely novel approach, namely, by bringing the user in the loop, and allowing him or her to give us some example records that he or she considers as outliers.

The contributions of this paper are the following:

– We propose OBE, which, to the best of our knowledge, is the first method to provide a solution to this problem.

- We build a system, and described our design decisions. Although OBE appears simple to the user ("click on a few outlier-looking records"), there are many technical challenges under the hood. We showed how to approach them, and specifically, how to extract suitable feature vectors out of our data objects, and how to quickly train a classifier to learn from the (few) examples that the user provides.
- We evaluated OBE on both real and synthetic data, with several small sets of user examples. Our experiments demonstrate that OBE can succesfully incorporate these examples in the discovery process and detect outliers with "outlier-ness" characteristics very similar to the given examples.

Acknowledgements. The authors are grateful to Dr. Phillp B. Gibbons for early discussion on example-based outlier detection. This research has been supported in part by Japan-U.S. Cooperative Science Program of JSPS, U.S.-Japan Joint Seminar (NSF grant 0318547) and the Grant-in-Aid for Scientific Research from JSPS and MEXT (#15300027, #15017207).

References

1. V. Barnett and T. Lewis. Outliers in Statistical Data. John Wiley and Sons, 1994.
2. 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.
3. S. D. Bay and M. Schwabacher. Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. In SIGKDD'03, August 24-27, 2003.
4. D.M. Hawkins. Identification of Outliers. Chapman and Hall, 1980.
5. T.Johnson, I. Kwok, and R.T. Ng. Fast computation of 2-dimensional depth contours. In Proc. KDD, pages 224-228, 1998.
6. A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. ACM Comp. Surveys, 31(3);264-323,1999.
7. E.M. Knorr and R.T. Ng. A unified notion of outliers: Properties and computation. In Proc. KDD, pages 219-222, 1997.
8. E.M. Knorr and R.T. Ng. Algorithms for mining distance-based outliers in large datasets. In Proc. VLDB 1998, pages 392-403, 1998.
9. E.M. Knorr and R.T. Ng. Finding intentional knowledge of distance-based outliers. In Proc. VLDB, pages 211-222, 1999.
10. E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. VLDB Journal, 8:237-253, 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. H. Yu, J. Han and K.Chang. PEBL: Positive Example Based Learning for Web Page Classification Using SVM. In Proc. KDD, 2002.
14. http://www.csie.nut.edu.tw/~cjlin/libsvm.
15. K. Yamanishi and J.Takeuchi. Discovering Outlier Filtering Rules from Unlabeled Data. In Proc. KDD, 2001.