Regular Paper

Example-Based Outlier Detection for High Dimensional Datasets

Cui Zhu,[†] Hiroyuki Kitagawa[†,††] and Christos Faloutsos[†††]

Detecting outliers is an important problem, in applications such as fraud detection, financial analysis, health monitoring and so on. It is typical of most such applications to possess high dimensional datasets. Many recent approaches detect outliers according to some reasonable, pre-defined concepts of an outlier (e.g., distance-based, density-based, etc.). Most of these concepts are proximity-based which define an outlier by its relationship to the rest of the data. However, in high dimensional space, the data becomes sparse which implies that every object can be regarded as an outlier from the point of view of similarity. Furthermore, a fundamental issue is that the notion of which objects are outliers typically varies between users, problem domains or, even, datasets. In this paper, we present a novel solution to this problem, by detecting outliers based on user examples for high dimensional datasets. By studying the behavior of projections of such a few outlier examples in the dataset, the proposed method discovers the hidden view of outliers and picks out further objects that are outstanding in the projection where the examples stand out greatly. Our experiments on both real and synthetic datasets demonstrate the ability of the proposed method to detect outliers that match users' intentions.

1. Introduction

Abnormal or outlier objects often contain useful information in applications like fraud detection, financial analysis and health monitoring. It is an interesting and important problem to detect such outliers in large datasets. However, it is also a difficult problem especially when the data is in high dimensional spaces, which is often the case with most its applications.

In general, an outlier is an object which is different greatly from the rest of the data.There are various interpretations of the notion of outlier in different scientific communities, based on some measure of difference like distance-based [14], density-based [9], etc. However, these definitions of an outlier which are based on proximity is not meaningful when the dimensionality is very high. In fact, all pairs of points are almost equidistant in a high dimensional space, for a wide range of data distributions and distance functions [12]. This infers that each object is an equal good outlier from a point of view of proximity.

It has been shown that by examining the behavior of the data in low dimensional projections, meaningful outliers are likely to be defined [2]. It is also observed that different objects may be detected as outliers with respect to different subsets of dimensions [2].

Naturally, the notion of what is an *outlier* 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).

Example

In order to clarify the point, we give an example illustrated in **Figs. 1** and **2**. The example is a real abalone dataset obtained from the UCI machine learning repository [13]. The abalone dataset has eight numerical features. In Fig. 1, the triangle objects present abnormal behavior in subspace1, i.e., diameter-whole weight subspace. However, most of the same objects show average behavior in other views like subspace2 (shucked weight-shell weight subspace). At the same time, outliers (circle dots) detected in subspace2 are overwhelmed in subspace1 in Fig. 2.

This is consistent with intuition that different objects may be regarded as outliers, depending on different views from which we examine the datasets.

Being experts in their problem domain, not in outlier detection, users often have a few example outliers in hand, which may "describe" their

† Graduate School of Systems and Information Engineering, University of Tsukuba
†† Center for Computational Sciences, University of Tsukuba
††† School of Computer Science, Carnegie Mellon University

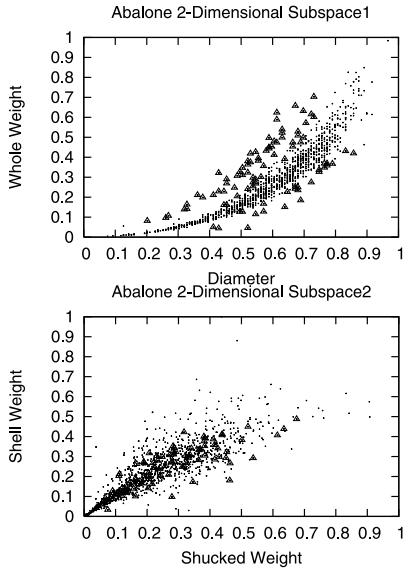


Fig. 1 Illustration of same objects in different subspaces for Abalone Outliers O-DW. Top: O-DW outliers in subspace1, bottom: O-DW outliers in subspace2.

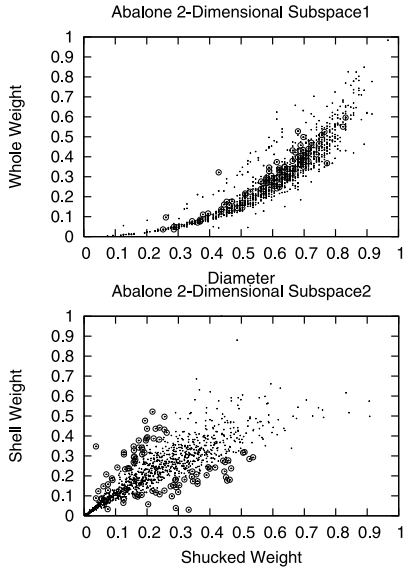


Fig. 2 Illustration of same objects in different subspaces for Abalone Outliers O-SS. Top: O-SS outliers in subspace1, bottom: O-SS outliers in subspace2.

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 to find out the "hidden" outlier concept that users may have in mind.

Several problems should be resolved to design an example-based outlier detection approach for high dimensional datasets. We briefly list the most important: **(1)** How to define "outlier-ness" meaningfully in a low dimensional subspace, if not in the original high dimensional space? **(2)** How to measure the degree of "outlier-ness" identically as well in subspaces with different dimensions. **(3)** The approach should provide some insights to the reasoning that brings about the abnormality. **(4)** It should be computationally efficient to detect outliers for high dimensional applications. **(5)** At last, the method should clearly require minimal user input and effectively use a *small* number of outlier examples in order to be practical. Given these requirements, can we design a method which uses only the handful of outlier examples and discovers more other objects with similar outlier characteristics?

In this paper, we propose an novel outlier detection method specially designed for high dimensional datasets that can discover the desired view of "outlier-ness", based on a small number of examples.

The remainder of the paper is organized as follows: In the next section, we discuss related work on outlier detection. In Section 3, we discuss the definition of "outlier-ness" in a high dimensional dataset. Section 4 presents the novel method to detect outliers based on examples and its design decisions in detail. The experiment evaluation on both synthetic and real datasets are reported in Section 5. 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 [7),15),19)]. They deploy some distribution models, objects which deviate from the model are flagged as outliers. However, they are unsuitable for high dimensional datasets. When dimensionality increased, it is difficult, expensive and inaccurate to assess the multidimensional distributions of points in the feature space. (2) *Depth-based approach.* This computes different layers of k-d convex hulls and flags objects in the outer layer as outliers [18)]. It is known that these algorithms also suffer from the curse of dimensionality. (3) *Clustering approach.* Many cluster-

ing algorithms detect outliers as by-products [8]. Obviously, they are not optimized for outlier detection. (4) *Distance-based approach.* Distance-based outliers [9)~11),17] are defined by using a full dimensional distances of the points from one another. When dimensionality is high and the data is sparse, the full dimensional distances become no more meaningful. (5) *Density-based approach* [14]. They introduced a local outlier factor (LOF) for each object, indicating its degree of "outlier-ness." LOF depends on the local density which is computed by using the distance to the MinPts-th nearest neighbor. It is again difficult to define the concept of locality meaningfully in high dimensionality due to data sparsity. (6) *LOCI.* Papadimitriou and others [16] proposed the multi-granularity deviation factor (MDEF) and LOCI. MDEF measures the "outlier-ness" of objects in neighborhoods of different scales. LOCI examines the MDEF values of objects and flags as outliers those objects whose MDEF values deviate significantly in neighborhoods of *some* scales. Our previous work proceeded to use MDEF to detect outliers based on user examples [20]. Basically, MDEF defines neighborhoods by full dimensional distances. Therefore, this measure of "outlier-ness" naturally fails to deal with high dimensionality.

In order to deal with the curse of high dimensionality, a quite different technique is proposed by Aggarwal and Yu [2], 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 GA algorithm, and all the objects in these cubes are reported as outliers. However, no notion of example was used in their approach. Furthermore, the dimensionality of projections is assumed to be given by users. Without any prior knowledge of the datasets, it is hard for users to figure out.

In summary, none of existing methods detect outliers by incorporating directly user examples and are specifically designed for the high dimensional datasets as well.

3. Definition of Outliers for High Dimensional Datasets

Researches in the contexts like clustering and similarity search [1),3),6)] show that it is possible to design more effective algorithms by studying the behavior of data in subspaces. This insight is similar for outlier detection. Aggarwal and Yu [2] defined outliers by examining projections of data whose density is abnormally low. "Outlier-ness" is measured by the sparsity coefficient shown below. In our approach to example-based outlier detection in high dimensional datasets, we employ their sparsity coefficient.

In order to define such projections, they divide each attribute of the data into φ equi-depth ranges. Then, in each range, there are a fraction $f = 1/\varphi$ of the data. Here equi-depth rather than equi-width ranges are employed because different localities may have different densities and such local density problem should be taken into account while detecting outliers.

A k-dimensional cube is made by ranges from k different dimensions. (Note that each range is associated with $f = 1/\varphi$ of the data.) The expected fraction of objects in the cube would be f^k, if the attributes were independent statistically. While attributes in real applications is far from statistically independent, the actual fraction of objects in a cube would differ greatly from the average behavior.

Let N be the dataset size and $n(D)$ denotes the number of objects in a k-dimensional cube D. If the attributes were independent statistically, then the number of points in a k-dimensional cube is a Bernoulli random variable with probability f^k of presence, because of the equi-depth grids. Consequently, the number of points in a cube will approximately follows a binomial distribution. Then the average number of objects in a k-dimensional cube is $N \cdot f^k$, and the standard deviation is $\sqrt{N \cdot f^k \cdot (1 - f^k)}$. Then the sparsity coefficient $S(D)$ of the cube D can be calculated by:

$$S(D) = \frac{n(D) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}} \qquad (1)$$

Negative sparsity coefficients indicate those cubes that contain less objects than average. The less the objects is in a k-dimensional cube, the smaller the sparsity coefficient. For the same number of objects in a cube, lower dimensionality of the cube leads to smaller sparsity coefficient[☆]. Even though the attributes may not be statistically independent, the sparsity coefficient implies the degree of deviation of density in a cube from the average.

[☆] We have proved that the partial derivative of $S(D)$ with respect to k is always great than 0, indicating that $S(D)$ is a monotonously increasing function of k.

At last, objects in the significantly sparse cubes are defined as outliers.

4. Proposed Method

In this section, we discuss the algorithm to detect outliers based on examples in high dimensional datasets. Here, the problem is whether we can discover a low dimensional subspace where the user examples are outstanding significantly than any other subspaces. And objects that are outlying in the same subspace seem to accord with the user's view of what is an outlier.

Let the total dimensionality of the data is d. In order to find a k-dimensional subspace where examples are isolated from the great majority, $\binom{d}{k}$ possible combinations of dimensions should be examined. With no idea of the dimensionality of the subspace, we have to check all subspaces when dimensionality vary from 2 to d. Totally, there are $\sum_{k=2}^{d}\binom{d}{k}$ possible combinations of dimensions to be examined. It is exhausting or even untenable to discover a desired subspace by brute force method which examines all these sets of possible combinations,even for middle d. For example, the search space is around 10^{14} for datasets of 50 dimensions.

We propose to solve the problem by employing an evolutionary algorithm which is able to quickly find the hidden subspace in which examples stand out greatly.

4.1 An Overview of Evolutionary Algorithm

Evolutionary algorithms [5] are developed on the principles of natural evolution in order to solve parameter optimization problems. These class of search methods model some natural phenomena like the Darwinian strife for survival and approximate an optimal solution to the problem at hand.

In the algorithms, each solution to an optimization problem can be represented as a string, called a chromosome. The coding strings are composed of features that are analogous to genes. Each feature has its own position and a definite value. The quality of a solution is estimated based on the "fitness" value, which is its objective function. Instead of a single solution, evolutionary methods work with a group of solutions called a population. The population undergoes repeated processes of selection, crossover and mutation. During the evolvement, members with fitter values are more likely to survive and participate in recombination operations. Mutation occasionally throws in a variant, so as to add to the diversity of a population. Thus, the methods search in a greater scope for improvement and the population becomes overall better and better until it converges and a "best" solution may be found. These processes combine the methods of hill climbing, solution recombination and random search over the search space. That is why evolutionary algorithms can often outperform classical methods of optimization when applied to difficult, real-world problems.

Evolutionary algorithms start with encoding a solution carefully for a given problem. Then genetic operators should be designed elaborately as well. These processes provide a way to incorporate domain knowledges into evolutionary algorithms and play key roles in determining their performances.

4.2 The Algorithm of Outlier Detection Based on Examples

Here we discuss the application of the evolutionary algorithm to the outlier detection problem.

In the context, a solution indicates a subspace. Therefore a bit string representation is used. The length of a coding string is equal to d, the total dimensionality of the data. Each position in the string corresponds to a dimension and 1 denotes that the corresponding dimension is identified with the subspace of the solution whereas 0 means not. For example, for a 4-dimensional problem, a solution of 1010 means that the subspace is constructed by the first and third dimensions.

The fitness of a solution is equal to the average value of sparsity coefficients of cubes in the subspace which contain user examples. The evolutionary search method starts with a population of p random solutions. Then it seeks to maximize the overall fitness function in the population by an iterative processes of selection, crossover and mutation, until the population converge to a global optimum. The convergence goal of a population is achieved when all features in solutions have converged. Convergence of a feature is defined as the stage where 95% of the population had the same feature value [4]. Note that in our context, to maximize the fitness value is achieved by minimizing the average sparsity coefficient.

Finally, the best solution discovered by the evolutionary algorithm is just the subspace

```
Input:
  Set of outlier examples: E
  Number of equi-depth ranges for each attribute: φ
  Dataset: DS
Output:
  Outliers: O
Algorithm:
begin
  P := Initial Population of p solutions;
  while not(convergence_criterion) do begin
    P := Selection(P);
    P := Crossover(P);
    P := Mutation(P);
    end;
  S :=Solution in P with the maximum fitness value;
  O :=Sets of objects contained in cubes in S which
    are sparser than or as sparse as cubes containing
    examples;
  return O;
end
```

Fig. 3 Overall procedure of the example-based outlier detection algorithm.

where user examples stand out significantly. Then to discover more outliers, we scan the subspace to find objects in cubes which are sparser than or as sparse as those of examples and report them as outliers.

Figure 3 illustrates the overall procedure of the algorithm. The evolutionary algorithm employs the following processes in order to create a new better population from the current generation.

(1) Selection

Selection process chooses parents for the next generation. The idea is that better solutions should have a larger number of copies. We use rank selection mechanism because it is often more stable. First all solutions are sorted in descending order of their fitness values. A line is laid out in which each solution corresponds to a section of the line of length proportional to its rank. The mechanism moves along the line in steps of equal size. At each step, it allocates a parent from the section it lands on. The first step is a uniform random number less than the step size. Thus a solution with better fitness (namely, smaller average sparse coefficients of cubes holding examples) will have more chances to be selected.

(2) Crossover

Crossover technique plays a key role in evolutionary algorithms to find a better child by combinations of parent solutions. First we explain the common scattered crossover mechanism and then show the optimized crossover method specially designed for our outlier detection problem.

Scattered Crossover It is one of the popularly used mechanisms in evolutionary algorithms to create the recombinant children strings. The scattered mechanism starts by creating a random binary vector. Then it selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent. The child is formed by combining the selected genes. For example, if p1 and p2 are the parents: p1 = 1100, p2 = 0011, and the binary vector is 1010, the recombined child is 1001.

Note that if the binary vector is 1011, the recombined child would be 1000. This string indicates a subspace formed only by the first dimension. In a 1-dimensional subspace, the data is uniformly allocated in all ranges because we use a equi-depth grid. This kind of string is useless to find outliers. Such solutions tend to be discarded in subsequent generations because of their worse objective values. Evolutionary algorithms have bad performance if the recombination process cannot create solutions of high quality. Therefore, we propose an optimized crossover process which avoids such strings and outputs better children.

Optimized Crossover Except positions where both parent strings have 0, there are two types of positions in the parent strings. If both parent strings have 1 in the position, it is type I. Otherwise, it is type II. It is obvious that the child string can only change within possible combinations of positions in type II. Of course, to find the best recombination child from the parents is the ideal goal. However, it is difficult to achieve since there are a total of $\sum_{k'=0}^{x} \binom{x}{k'}$ possibilities for the child string, when there are x positions of type II. In order to make the crossover operation effective, we look for a child string that has better fitness value.

We proved that the sparsity coefficient $S(D)$ is a monotonously increasing function of k, when $k > 1$ and $\varphi > 1$. This implies that a better solution, i.e., a smaller $S(D)$ value is likely to be found in lower dimensional subspace. Therefore, we examine the set of possible recombination from low dimensionality. The recombination with lowest dimensionality is a solution whose values are 0 in all po-

```
Input:
  Two parent strings: p1, p2
Output:
  A child string: child
Algorithm:
begin
  T1 := Set of positions where p1 = p2 = 1;
  T2 := Set of positions where p1 ≠ p2;
  s := Solution(T1);
  while s is worse than p1 and p2 do begin
    for each V_i ∈ T2
      Q_i := Solution(V_i ∪ T1);
    s := Best solution in all the Q_i;
    T2 := T2-(1positions(s)-T1);
    T1 := 1positions(s);
    end;
  child :=s;
  return child;
end
Algorithm: Solution(T)
begin
  return A solution whose values of positions
    in T are 1 and 0 for the rest.
end
Algorithm: 1positions(s)
begin
  return A set of positions where the values are 1 in s;
end
```

Fig. 4 Optimized crossover algorithm.

sitions of type II. We examine such a solution first and keep increasing the dimensionality of the solution by adding one position from type II, which results in the solution with minimal average sparsity coefficient, until a solution that is better than the two parent solutions is found. Then, the better solution is returned as the child recombination. Such a crossover procedure creates a new solution which combines the good aspects of both parent strings. The optimized crossover algorithm is shown in **Fig. 4**.

(3) Mutation

A simple uniform mutation algorithm is used. It is a two-step process. First, the algorithm picks a fraction of the positions of a solution for mutation, where each position has a probability *prob* of being mutated. In the second step, the algorithm inverses each selected position to form a child solution.

4.3 Postprocessing Phase

After the evolutionary algorithm terminates, the subspace in which user examples are outlying greatly is found. In the postprocessing phase, we find all the objects contained in the cubes which are sparser than or as sparse as cubes containing examples in the subspace.

Table 1 Description of synthetic and real datasets. Dim denotes dimensionality of the dataset.

Dataset	Dim	Description
Data I	20	20,100 data which are uniformly distributed in 16 dimensions and holding two sets of outliers in the remainder two 2-dimensional subspaces.
Data II	30	20,100 data. 25 dimensions are uniformly distributed. There are two sets of outliers obtained in 3-dimensional and 2-dimensional subspaces respectively.
Abalone	8	Abalone data, obtained from the UCI machine learning repository, 4177 examinations of abalones with 8 attributes.

These objects are example-like outliers and are reported to users.

5. Experimental Evaluation

In this section, we describe our experimental methodology and the performance of the proposed method over synthetic and real data. These experiments demonstrate effectiveness and efficiency of our method.

5.1 Datasets

We test the proposed method on two synthetic and one real dataset (see **Table 1** for descriptions). In the low dimensional subspaces of synthetic data, we let the majority distributed uniformly in some areas and scatter a few isolated outliers in other zones. For each test data, two sets of outlier examples with quite different natures are engaged to test the capability of the method of detecting outliers from different views of users.

5.2 Experimental Procedure

Our experimental procedure is as follows:

(1) To label low dimensional outliers, we select objects which are in the most sparse cubes in the relevant subspaces. More concretely, we divide each attribute of the data into φ equi-depth ranges. Several ranges of diverse attributes form a cube. If the number of objects in a cube is small than θ (θ is a threshold), we define all objects in the cube as outliers.

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

(3) Next, we detect outliers using the proposed method, by learning only from

☆ In our experiments, $y = 10$.

Table 2 Interesting outliers and the definitions. S-Dim denotes dimensionality of the relevant subspace.

Dataset	Outlier Description				
	Label	S-Dim	φ	θ	# of Outliers
Data I	O-2D-1	2	20	4	99
	O-2D-2	2	20	4	146
Data II	O-3D-1	3	10	2	74
	O-2D-2	2	20	4	83
Abalone	O-DW	2	20	4	89
	O-SS	2	20	4	88

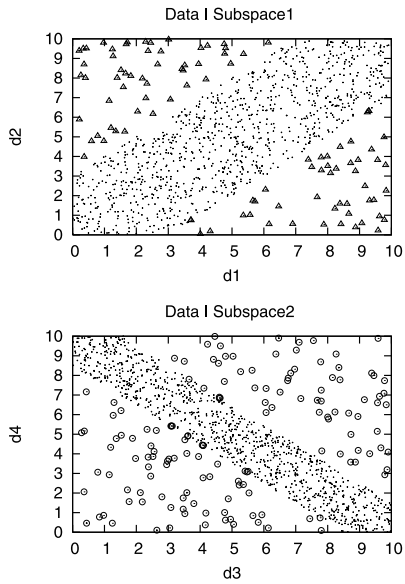


Fig. 5 Outliers in low-dimensional subspaces of data I. Top: O-2D-1 outliers in subspace1, bottom: O-2D-2 outliers in subspace2.

such a few examples.

(4) We mainly test two aspects of our method: one is how often the evolutionary algorithm discovers the right relevant subspaces, and the other is the execution time.

Description of the outliers and their definitions is shown in **Table 2**. **Figure 5** displays two subspaces of data I where outliers are discovered. Corresponding subspaces of data II are shown in **Fig. 6**. For abalone data, Figs. 1 and 2 in Section 1 illustrate two kinds of outliers which are discovered respectively in diameter-whole weight and shucked weight-shell weight subspaces.

In order to show how few the examples are, we display two sets of the picked examples in the abalone data in **Fig. 7**. One of picked O-DW outliers in Fig. 7 has the diameter of 0.521008, but weights only 0.088496. However the majority of the like diameter weight from 0.15 to over 0.2. Other users may give an abalone example whose weight of meat (i.e. shucked weight) is 0.332213 and having an extremely light shell of 0.029895. These abnormal abalones are called O-SS outliers in our paper.

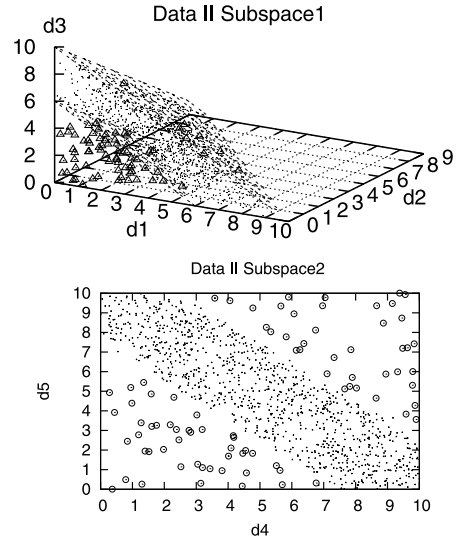


Fig. 6 Outliers in low-dimensional subspaces of data II. Top: O-3D-1 outliers in subspace1, bottom: O-2D-2 outliers in subspace2.

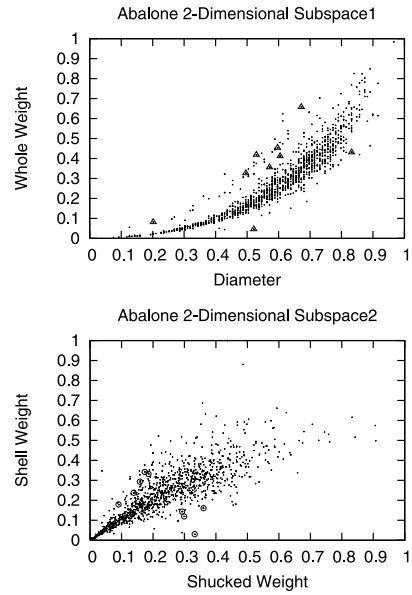


Fig. 7 Two sets of examples for abalone. Top: O-DW examples in subspace1, bottom: O-SS examples in subspace2.

5.3 Results

The algorithm is implemented on a 2.80 GHz machine of Windows XP with 1024 MB of main

Table 3 Average # of iterations, accuracy and time showing performance of the scattered and optimized crossovers. Note that all the results are average of 100 trials from different sets of examples and initial population.

Test Data		Scattered Crossover			Optimized Crossover		
		Average # of iters	Accuracy	Time (Second)	Average # of iters	Accuracy	Time (Second)
Data I	O-2D-1	16.4	64%	124	5.6	96%	228
	O-2D-2	16.6	65%	186	5.6	93%	349
Data II	O-3D-1	20.5	37%	159	7.2	84%	244
	O-2D-2	19.8	31%	148	6.7	71%	178
Abalone	O-DW	11.4	51%	3	4.4	98%	5
	O-SS	11.4	51%	3	4.4	99%	5

memory. We particularly compare the performance of the scattered and optimized crossover mechanisms. The evolutionary algorithm is run on each test data 100 times with different sets of examples and initial populations. Statistical measurements of accuracy are obtained on success rate of the methods in discovering the right subspaces among 100 trials.

Table 3 shows the results. It is obvious that the optimized crossover mechanism outperforms substantially in terms of the accuracy of finding the best solution. This is because the optimized crossover process always identifies better child solutions by combining the good aspects of both parent solutions.

The average number of iterations required for convergence is shown under the column (Average # of iters). Time results are also the average of 100 trials. As expected, although the optimized crossover process needs less number of iterations to converge, they spend a little more time than the scattered crossover method. The reason is that in each process of the optimized crossover, at least one possible recombination is evaluated in order to find a child solution better than both parent solutions. Note that the time extension is within a reasonable and tolerant range.

When we increase the size of the population in the evolutionary algorithm, more time will be required to converge. On the other hand, the increased population size tends to discover the best solution with higher probability. **Figure 8** shows the performance of the proposed methods in the terms of accuracy versus time consuming. Results in Fig. 8 are obtained when we test the scattered and optimized crossovers over a set of various population sizes.☆ It is evident from these results that the optimized crossover process works qualitatively quite better than the plain scattered crossover for test data.

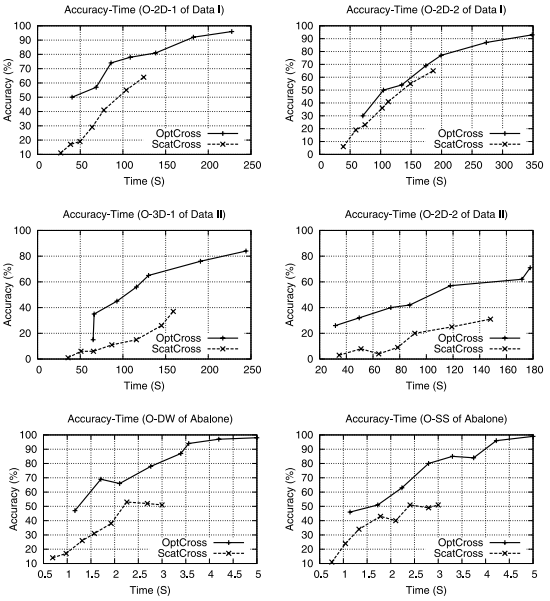


Fig. 8 Accuracy and time. Top left: O-2D-1 in data I, top right: O-2D-2 in data I, middle left: O-3D-1 in data II, middle right: O-2D-2 in data II, bottom left: O-DW in abalone data, bottom right: O-SS in abalone data.

5.4 Effect of Examples

Now that there exist subjectivities of outlier notion, by incorporating user examples directly in detecting process, our method discovers outliers which are more likely to meet user's interests. Other methods which leave users outside the loop of detection do not have this merit.

Specifically, when we run Aggarwal's algorithm in Ref. 2) on data I, all 245 outliers including 99 O-2D-1 and 146 O-2D-2 are found together. (Note that there are no more 2-dimensional outliers except O-2D-1 and O-2D-2 in data I.) Using Aggarwal's algorithm, it is up to users to evaluate those reported outliers

☆ In our experiments, the population size is varied as 20, 30, 40, 50, 60, 80 and 100.

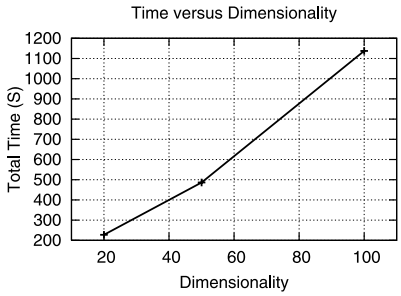


Fig. 9 Time cost versus number of dimensionality.

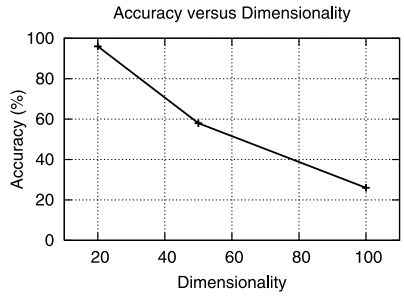


Fig. 10 Accuracy versus number of dimensionality.

manually. However our method detects those outliers similar to user examples. Based on a few O-2D-1 examples (only 9 as 10%), for example, only O-2D-1 outliers are discovered and reported.

For abalone data, things become worse. Aggarwal's algorithm finds a total of 269 best outliers at one time. Among them, only 40 abalones are outliers in O-SS and 69 outliers belong to O-DW. Note also that the results are obtained under the condition that the dimensionality of subspaces is determined correctly and a proper number of best cubes is given as well.☆ However, our method does not need these two inputs and only needs a few outlier examples.

5.5 Impact of Dimensionality

To answer the question of how high dimensional data our method can cope with, we conduct more experiments. Additional test data of 50 and 100 dimensions are constructed by adding extra attributes of uniform distribution to the data of data I. Also 10% of O-2D-1 outliers are sampled as examples. We always set population size as 100 and examine the performance of optimized crossover algorithm (as aforementioned, optimized crossover performs better in our experiments). We report accuracy rate and time required when population converged.

Results are shown in **Figs. 9** and **10**. Total time in Fig. 9 includes CPU and I/O taken to detect outliers and is average of 50 trials. Accuracy is success rate for the method discovering the right subspace among 50 trials.

As expected, when the dimensionality increases the problem becomes difficult and more time is required for convergence of population. Probability of success decreases for higher dimensional data. In order to increase probability of success, we have to raise the population size. And this in turn results in more computational time, just as experiments in Fig. 8 have proven. Obviously, it is a tradeoff of time and accuracy.

☆ The Aggarwal outliers of abalone are detected in cubes with best two sparsity coefficients.

6. Conclusion

It is an important and difficult problem to detect outliers in high dimensional data. Besides, the problem is tricky, since the exact notion of an outlier often depends on the user and/or the dataset. We proposed to solve this problem by detecting outliers based on user examples which indicate the user's perspective of what is an outlier. The proposed method is especially designed to deal with the curse of dimensionality.

The method works by examining the behavior of examples in low-dimensional subspace and finds one in which such examples are isolated from the great majority. This search problem cannot be solved by the brute-force method due to the number of possible combinations. A evolutionary algorithm is designed to address the search problem. Experiments on both real and synthetic data strongly indicate that the method can find the best relevant subspaces from the standpoints of users with high confidence and within reasonable time.

References

1) Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P.: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *Proc. SIGMOD Conf.*, pp.94–105 (1998).
2) Aggarwal, C.C. and Yu, P.S.: Outlier Detection for High Dimensional Data, *Proc. SIGMOD Conf.* (2001).
3) Aggarwal, C.C. and Yu, P.S.: Finding Generalized Projected Clusters in High Dimensional

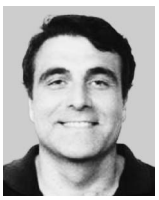
Spaces, *Proc. SIGMOD Conf.*, pp.70–81 (2000).
4) De Jong, K.A.: Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI (1975).
5) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1989).
6) Hinneburg, A., Aggarwal, C.C. and Keim, D.A.: What is the Nearest Neighbor in High Dimensional Spaces? *Proc. VLDB*, pp.506–515 (2000).
7) Hawkins, D.M.: *Identification of Outliers*, Chapman and Hall (1980).
8) Jain, A.K., Murty, M.N. and Flynn, P.J.: Data Clustering: A Review, *ACM Comp. Surveys*, Vol.31, No.3, pp.264–323 (1999).
9) Knorr, E.M. and Ng, R.T.: Algorithms for Mining Distance-Based Outliers in Large Datasets, *Proc. VLDB*, pp.392–403 (1998).
10) Knorr, E.M. and Ng, R.T.: Finding Intentional Knowledge of Distance-Based Outliers, *Proc. VLDB*, pp.211–222 (1999).
11) Knorr, E.M., Ng, R.T. and Tucakov, V.: Distance-Based Outliers: Algorithms and Applications, *VLDB Journal*, Vol.8, pp.237–253 (2000).
12) Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U.: When is Nearest Neighbors Meaningful? *Proc. Int. Conf. Database Theories*, pp.217–235 (1999).
13) http://www.ics.uci.edu/˜ mlearn/ MLRepository.html
14) Breunig, M.M., Kriegel, H.P., Ng, R.T. and Sander, J.: LOF: Identifying Density-Based Local Outliers, *Proc. SIGMOD Conf.*, pp. 93–104 (2000).
15) Rousseeuw, P.J. and Leroy, A.M.: *Robust Regression and Outlier Detection*, John Wiley and Sons (1987).
16) Papadimitriou, S., Kitagawa, H., Gibbons, P.B. and Faloutsos, C.: LOCI: Fast Outlier Detection Using the Local Correlation Integral, *Proc. ICDE*, pp.315–326 (2003).
17) Bay, S.D. and Schwabacher, M.: Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule, *Proc. KDD* (2003).
18) Johnson, T., Kwok, I. and Ng, R.T.: Fast Computation of 2-Dimensional Depth Contours, *Proc. KDD*, pp.224–228 (1998).
19) Barnett, V. and Lewis, T.: *Outliers in Statistical Data*, John Wiley and Sons (1994).
20) Zhu, C., Kitagawa, H., Papadimitriou, S. and Faloutsos, C.: OBE: Outlier by Example, *Proc. PAKDD*, pp.222–234 (2004).

(Received December 20, 2004)
(Accepted April 6, 2005)

(Editor in Charge: *Atsuhiro Takasu*)

Cui Zhu is a student of Graduate School of Systems and Information Engineering, University of Tsukuba. She received the M.Sc. degree from School of Mechanical Engineering & Automation, Beihang University, China, in 1999. Her research interests include data and web mining. She is a student member of ACM, ACM SIGMOD Japan, and DBSJ.

Hiroyuki Kitagawa is a Professor at Graduate School of Systems and Information Engineering, University of Tsukuba. His research interests include integration of heterogeneous information sources, WWW and databases, structured documents, semi-structured data, multimedia databases, and human interface. He is a member of ACM, IEEE Computer Society, DBSJ, IEICE, IPSJ, and JSSST.



Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), four "best paper" awards, and several teaching awards. He is a member of the executive committee of SIGKDD; he has published over 120 refereed articles, one monograph, and holds four patents. His research interests include data mining for streams and networks, fractals, indexing methods for spatial and multimedia bases, and data base performance.