Knowl Inf Syst (2011) 28:709–733
DOI 10.1007/s10115-010-0323-y

REGULAR PAPER

COID: A cluster–outlier iterative detection approach to multi-dimensional data analysis

Yong Shi · Li Zhang

Received: 25 July 2008 / Revised: 12 April 2010 / Accepted: 28 June 2010 /
Published online: 11 July 2010

Abstract Nowadays, most data mining algorithms focus on clustering methods alone. Also, there are a lot of approaches designed for outlier detection. We observe that, in many situations, clusters and outliers are concepts whose meanings are inseparable to each other, especially for those data sets with noise. Thus, it is necessary to treat both clusters and outliers as concepts of the same importance in data analysis. In this paper, we present our continuous work on the *cluster–outlier iterative detection* algorithm (Shi in SubCOID: exploring cluster-outlier iterative detection approach to multi-dimensional data analysis in subspace. Auburn, pp. 132–135, 2008; Shi and Zhang in Towards exploring interactive relationship between clusters and outliers in multi-dimensional data analysis. IEEE Computer Society. Tokyo, pp. 518–519, 2005) to detect the clusters and outliers in another perspective for noisy data sets. In this algorithm, clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached. This data processing algorithm can be applied in many fields, such as pattern recognition, data clustering, and signal processing. Experimental results demonstrate the advantages of our approach.

Keywords Clustering · Outlier detection · Multi-dimensional data ·
Cluster and outlier diversity

1 Introduction

The generation of multi-dimensional data has proceeded at an explosive rate in many disciplines with the advance in modern technology. Many new clustering, outlier detection,

Y. Shi (✉)
Department of Computer Science and Information Systems,
Kennesaw State University, Kennesaw, GA 30144, USA
e-mail: yshi5@kennesaw.edu

L. Zhang
Department of Computer Science, Eastern Michigan University, Ypsilanti, MI, USA

and cluster evaluation approaches have been presented in the last few years. Currently, many real data sets are noisy, which makes it more difficult to design and process algorithms efficiently and effectively.

We observe that, in many situations, clusters and outliers are concepts whose meanings are inseparable to each other, especially for those data sets with noise. Thus, it is necessary to treat clusters and outliers as concepts of equal importance in data analysis. In this paper, we present our continuous work on the *cluster–outlier iterative detection* (simplified as *COID* in the following content) algorithm [33,35] for noisy multi-dimensional data sets. In the previous works [33,35], we defined the problem and the basic terms for the algorithm and briefly described the algorithm. In this paper, we improve the definitions, present the details of the algorithm, and demonstrate the extensive experiments for the algorithm.

In the data analysis field, we observe another fundamental problem. Clusters and outliers are detected mostly based on the features of data sets, and the results are compared to the ground truth of natural clusters and outliers. However, in many cases, the available features do not reflect the ground truth very well. Also, good results are hard to achieve even using dimension reduction approaches. This is another motivation for us to develop the COID algorithm. We detect clusters and outliers in another perspective: not only relying mainly on the features of the data sets, but also exploiting the relationship between clusters and outliers in a computable way.

The remainder of this paper is organized as follows. Section 2 introduces the related work of clustering, outlier detection, and cluster evaluation. Section 3 presents the formalization and definitions of the problem. Section 4 describes the cluster–outlier iterative detection (*COID*) algorithm. Section 5 presents experimental results, and concluding remarks are offered in Sect. 6.

2 Related work

More and more large quantities of multi-dimensional data sets need to be clustered and analyzed. Many data mining approaches have been designed and analyzed [1,28,38,41,42]. Cluster analysis is applied to identify homogeneous and well-separated groups of objects in data sets. It plays an important role in many fields of business and science. The basic steps in the development of a clustering process can be summarized as data cleaning, feature selection, application of a clustering algorithm, validation of results, and interpretation of the results [14]. Among these steps, the clustering algorithm and validation of the results are especially critical, and many methods have been proposed in the literature for these two steps. Existing clustering algorithms can be broadly classified into four types [22]: partitioning [11,26,24,27], hierarchical [17,18,46], grid-based [3,32,39], and density-based [6,13,19] algorithms.

Partitioning algorithms construct a partition of a database of n objects into a set of K clusters, where K is an input parameter. In general, partitioning algorithms start with an initial partition and then use an iterative control strategy to optimize the quality of the clustering results by moving objects from one group to another. Hierarchical algorithms create a hierarchical decomposition of the given data set of data objects. The hierarchical decomposition is represented by a tree structure, called dendrogram. Grid-based algorithms quantize the space into a finite number of grids and perform all operations on this quantized space. Grid-based algorithms have the advantage of fast processing time independent of the data set size. The processing time is dependent only on the number of segments of each dimension in the quantized space. Density-based algorithms are designed to discover clusters of arbitrary shapes.

They hold that, for each point within a cluster, the density in the neighborhood of a given radius must exceed a defined threshold. Density-based approaches can also filter out outliers.

An outlier is a data point that does not follow the main characteristics of the input data. Outlier detection is concerned with discovering the exceptional behaviors of certain objects. It is an important branch in the field of data mining with numerous applications, including credit card fraud detection, discovery of criminal activities, discovery of computer intrusion, etc. In many applications, outlier detection is at least as significant as cluster detection. There are numerous studies on outlier detection [29,37,40,44]. Yu et al. [45] proposed an outlier detection approach termed *FindOut* as a by product of WaveCluster [32], which removes the clusters from the original data and identifies the outliers. Knorr and Ng [25] detected a distance-based outlier defined as a data point with a certain percentage of the objects in the data set, which have distances of more than $d_{\min}$ away from it. Ramaswamy et al. [30] further extended Knorr's work based on the distance of a data point from its kth nearest neighbor and identified the top n points with largest kth nearest neighbor distances as outliers. Breunig et al. [10] introduced the concept of *local outlier* and defined *local outlier factor* (LOF) of a data point as a degree of how isolated the data point was with respect to the surrounding neighborhood. Aggarwal and Yu [2] extended the problem of outlier detection in subspace to overcome *the curse of dimensionality*.

Our approach (COID) is different from previous clustering and outlier detection methods. We tried to detect and adjust the set of clusters and outliers according to both the intra-relationship and the inter-relationship between the set of clusters and the set of outliers. The related work was also presented in [34], which describes our initial attempt to solve the problem of clustering and outlier detection in subspace. In [34], we discussed the issue related to subspace-based approach, in which each cluster is associated with a unique subset of dimensions, so is each outlier. In each iteration, the subset of dimensions and the quality of each cluster are modified and adjusted dynamically, so are the subset of dimensions and the quality of each outlier. On the other hand, our approach (COID) focuses on the full data space analysis.

There are several criteria for quantifying the similarity (dissimilarity) of the clusters. ROCK [18] measured the similarity of two clusters by comparing the aggregate inter-connectivity of them. Chameleon [23] measured the similarity of two clusters based on a dynamic model. In the model, two clusters are merged only if the inter-connectivity and closeness (proximity) between the two clusters are highly related to the internal inter-connectivity of the clusters and closeness of items within the clusters.

Many approaches [12,20,43] have been proposed for evaluating the results of a clustering algorithm. Halkidi and Vazirgiannis [20] presented a clustering validity procedure. It defined a validity index, which contained the information of the average degree of scatter within clusters and the average number of points between the clusters. The index incorporated criteria addressing compactness, separation, and density. Chen and Lee [12] introduced a fuzzy validity function to measure the overall average compactness and separation of the fuzzy partition. The average compactness was estimated by the deviation of data points from the center of each cluster. The separation of the partition was represented by the distance between cluster centers. These clustering validity measurements evaluated clustering algorithms by measuring the overall quality of the clusters.

3 Problem definition

The concepts of cluster and outlier are related to each other. One aspect of the qualities of clusters and outliers is reflected by how much *diversity* they have inside and among each other. The cluster–outlier iterative detection (COID) problem is formalized as follows. Let n denote the total number of data points and d be the dimensionality of the data space. Let the input d-dimensional data set be $\mathbf{X} = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_n\}$, which is normalized to be within the hypercube $[0, 1]^d \subset R^d$. Each data point $\vec{X}_i$ is then a d-dimensional vector: $\vec{X}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$.

Based on the initial cluster–outlier division of a data set, the COID algorithm is performed iteratively. In one given iterative step, let the current number of clusters be k_c and the current number of outliers be k_o; let the set of clusters be $\mathcal{C} = \{C_1, C_2, \ldots, C_{k_c}\}$ and the set of outliers be $\mathcal{O} = \{O_1, O_2, \ldots, O_{k_o}\}$. We use the term *compactness* to measure the quality of a given cluster based on the closeness of its data points to its centroid.

3.1 Distance metric selection

It is crucial to define a proper distance metric for a specific data mining problem. We use $d(p_1, p_2)$ to represent the distance between data points p_1 and p_2 under a certain distance metric. In a high-dimensional space, data points are usually sparse, and widely used distance metrics such as Euclidean distance may not work well as dimensionality goes higher.

Recent research [8] shows that in high dimensions nearest neighbor queries become unstable: the difference of the distances of farthest and nearest points to some query point does not increase as fast as the minimum of the two. Aggarwal et al. [4,21] viewed the curse of dimensionality from distance metrics point of view. They argued that for the commonly used L_K norm, the problem of meaningfulness in high dimensionality was sensitive to the value of K: for high values of K, it worsens faster with increasing dimensionality. In this case, the Manhattan distance metric (L_1 norm) is consistently more preferable than the Euclidean distance metric (L_2 norm) for high-dimensional data mining applications. They also exploited the research in fractional distance metrics [4]. The fractional distance metric provides more meaningful results from both the theoretical and empirical perspective. It can significantly improve the effectiveness of standard clustering algorithms such as the k-means algorithm. Based on the current research, we prefer the fractional distance metric $L_{0.1}$ to L_2 metric in our implementations. For two data points $\vec{X}_1, \vec{X}_2 \in R^d$, $L_{0.1}(\vec{X}_1, \vec{X}_2) = \sum_{i=1}^{d}((x_{1i} - x_{2i})^{0.1})^{10}$.

3.2 The compactness of a cluster

A cluster in a data set is a subset where the included data points have a closer relationship with each other than with points outside the cluster. In the literatures [12,20], the intra-cluster relationship is measured by *compactness*, and the inter-cluster relationship is measured by *separation*. Compactness is a relative term; an object is compact in comparison to a looser surrounding environment.

Definition 1 Given the current set of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_{k_c}\}$ and the current set of outliers $\mathcal{O} = \{O_1, O_2, \ldots, O_{k_o}\}$, the **compactness** (CPT) of a cluster C_i is the closeness measurement of the data points in C_i to the centroid of C_i:

$$\text{CPT}(C_i) = \frac{\sum_{\vec{p} \in C_i} d(\vec{p}, m_{c_i})}{|C_i|}, \tag{1}$$

where m_{c_i} is the centroid of Cluster C_i, $\vec{p}$ is any data point in Cluster C_i, $|C_i|$ is the number of data points in C_i, and $d(\vec{p}, m_{c_i})$ is the distance between $\vec{p}$ and m_{c_i}. The centroid m_{c_i} of the cluster is the algebraic average of all the points in the cluster: $m_{c_i} = \sum_{\vec{p} \in C_i} \vec{p}/|C_i|$ where $|C_i|$ is the size of cluster C_i.

3.3 The diversities of data groups

We use the term *diversity* to describe the dissimilarity between two clusters, the dissimilarity between two outliers, and the one between a cluster and an outlier. We adopt *compactness* (CPT) concept (definition 1) of the cluster to set up the weights for the distance measurement.

Definition 2 The diversity between a cluster C and an outlier O is defined as:

$$D_1(C, O) = w_1 \cdot d_{\min}(O, C) + w_2 \cdot d_{\mathrm{avr}}(O, C) \tag{2}$$

where $w_1 = \frac{1}{\mathrm{CPT}(C)+1}$, $w_2 = \frac{\mathrm{CPT}(C)}{\mathrm{CPT}(C)+1}$, $d_{\mathrm{avr}}(O, C) = d(O, m_c)$, and $d_{\min}(O, C) = max(d(O, m_c) - r_{\max}, 0)$ where $r_{\max}$ is the maximum distance of the data points in C from its centroid.

In this way, the more compact the cluster is, the more important role $d_{\min}$ plays in determining the diversity between an outlier and a cluster. The larger the value of $D_1(C, O)$ is, the larger diversity the cluster C and the outlier O have to each other. If $1 \gg$ CPT(C), cluster is very compact: all the data points in C are extremely close to the centroid of C. In this case, $w_1 = \frac{1}{\mathrm{CPT}(C)+1} \approx 1$ and $w_2 = \frac{\mathrm{CPT}(C)}{\mathrm{CPT}(C)+1} \approx 0$; thus, $d_{\min}$ plays the major role in Definition 2. Actually, it is similar to the situation of calculating the diversity (distance) between two data points.

The distance between two clusters may not always represent their diversity accurately. Figure 1 shows an example of the relationship between clusters. Both clusters C_{11} and C_{12} are sparser than clusters C_{21} and C_{22}. From global point of view, the cluster pair (C_{11} and C_{12}) has a higher chance of being merged into a larger cluster than the pair (C_{21} and C_{22}). However, neither their minimum distance nor their average distance is different from each other ($d_{\min}(C_{11}, C_{12}) = d_{\min}(C_{21}, C_{22})$ and $d_{\mathrm{avr}}(C_{11}, C_{12}) = d_{\mathrm{avr}}(C_{21}, C_{22})$). There are also cases where the pair of clusters have different compactness and sizes as shown in Fig. 2, which makes the analysis even more complicated ($d_{\min}(C_{11}, C_{22}) = d_{\min}(C_{11}, C_{12}) = d_{\min}(C_{21}, C_{22})$ and $d_{\mathrm{avr}}(C_{11}, C_{22}) = d_{\mathrm{avr}}(C_{11}, C_{12}) = d_{\mathrm{avr}}(C_{21}, C_{22})$).

We apply a simplified criterion that integrates the *compactness* (CPT) concept (definition 1) into the diversity measurement of two clusters. It represents how compact the data points inside the cluster are.

Definition 3 *The diversity between two clusters C_1 and C_2 is defined as:*

$$D_2(C_1, C_2) = \frac{d(C_1, C_2)}{\mathrm{CPT}(C_1) + \mathrm{CPT}(C_2)} \tag{3}$$

where $d(C_1, C_2)$ can be either the average distance between the two clusters or the minimum distance between them. We just simply apply the former one $d(m_{C_1}, m_{C_2})$. The larger the value of $D_2(C_1, C_2)$ is, the larger diversity the clusters C_1 and C_2 have to each other.

Among the cluster pairs shown in Fig. 1 and Fig. 2, the cluster pair C_{11} and C_{12} will be chosen to merge prior to the cluster pair C_{21} and C_{22} or the pair C_{11} and C_{22}, although they are more sparse than the other two pairs. The reason is that $D_2(C_{11}, C_{12}) < D_2(C_{11}, C_{22}) < D_2(C_{21}, C_{22})$, according to Definition 3.

Definition 4 The diversity between two outliers O_1 and O_2 is defined as:

$$D_3(O_1, O_2) = d(O_1, O_2) \tag{4}$$

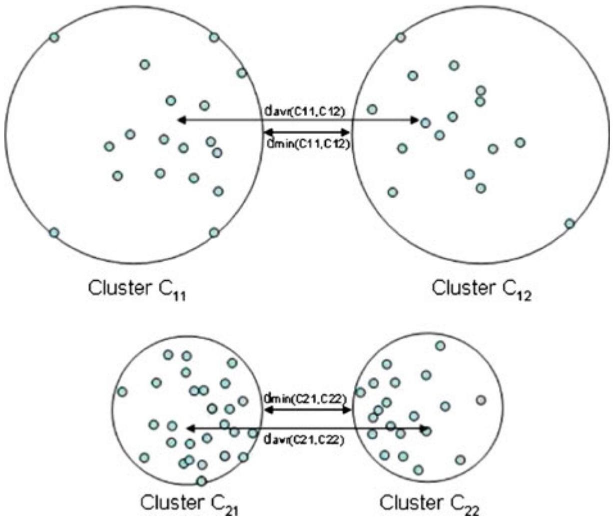

Fig. 1 Two cluster pairs with same average distance and minimum distance but different diversity

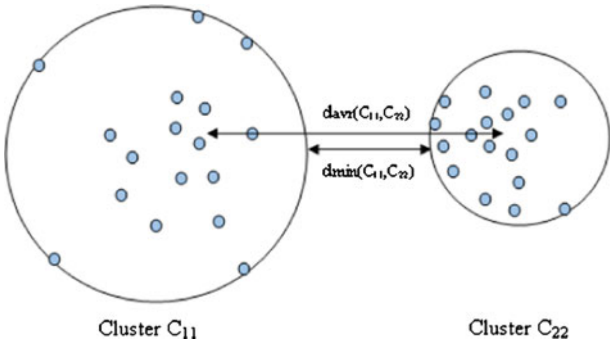

Fig. 2 A cluster pair with different compactness and sizes

3.4 The qualities of data groups

We propose a novel way to define the quality of a cluster C and an outlier O. The quality of C is reflected by the diversity between itself and other clusters (how far away and different they are from each other). It is also reflected by the diversity between C and outliers. If C is near some outliers, its quality should certainly be affected because outliers are supposed to be far away from any cluster. Thus, we take consideration of both the diversity between clusters and the diversity between a cluster and an outlier to define the quality of a cluster. We define the quality of an outlier O in a similar way.

Definition 5 We measure the quality of a cluster C as:

$$Q_c(C) = \frac{\frac{\sum_{C\prime \in \mathcal{C}, C\prime \neq C} D_2(C, C\prime)}{k_c - 1} + \frac{\sum_{O \in \mathcal{O}} D_1(C, O)}{k_o}}{\mathrm{CPT}(C)} \tag{5}$$

The larger $Q_c(C)$ is, the better quality cluster C has.

Definition 6 We measure the quality of an outlier O as:

$$Q_o(O) = \frac{\sum_{O\prime\in\mathcal{O}, O\prime\neq O} D_3(O, O\prime)}{k_o - 1} + \frac{\sum_{C\in\mathcal{C}} D_1(C, O)}{k_c} \tag{6}$$

The larger $Q_o(O)$ is, the better quality outlier O has.

4 Algorithm

The main objective of the *COID* algorithm is to mine the meaningful set of clusters and outliers from the input data set. Clusters and outliers are closely related and affect each other in a certain way. As mentioned in the previous sections, in our approach, clusters and outliers of multi-dimensional data are detected, adjusted, and improved iteratively. The overall pseudocodes for the algorithm are given in Fig. 3. Section 5.3 illustrates the entire process in detail.

The algorithm proceeds in two phases: an *initialization* phase and an *iterative* phase. In the *initialization* phase, we find the centers of clusters and locations of outliers. The problem of finding cluster centers has been extensively investigated. For example, in [9,15], the authors proposed a procedure for computing a refined starting condition from a given initial one that is based on an efficient technique for estimating the modes of a distribution. We apply a strategy similar to the greedy technique proposed in [16], which tries to iteratively choose surrogate cluster centers ($medoids$). In the *iterative* phase, we refine the set of clusters and outliers gradually by exchanging the outliers of the worst qualities with the boundary data points of the worst clusters. Each phase is detailed in the following.

4.1 Initialization phase

In the initialization phase, first, the initial set of medoids is found. Next, data points are dispatched to their nearest medoids, forming data subsets associated with the medoids. Approaches are then exploited to determine whether a data subset is a cluster or a group of outliers.

4.1.1 Acquirement of medoids

Finding medoids that can approximate the centers of different clusters is critical for our approach. Since outliers normally exist in real data sets, it is not effective to apply the greedy technique of [16] directly to find k medoids. Similar to the method proposed in [5], we try to find a *superset* of a good set of k medoids. We also develop a novel approach to determine the set of k medoids from its superset.

We first choose a random set RS_1 of data points from the original data set. RS_1 has the size `RandomSize1`, which is multiple of the required cluster number k. We then apply the greedy algorithm in [16] to find another random set RS_2 from RS_1. To form RS_2, each time we find a data point from RS_1, which is farthest to its closest data point among all the current data points in RS_2, and put it into RS_2, until the size of RS_2 reaches the value `Random-Size2`. `RandomSize2` is also multiple of k, and `RandomSize1 > RandomSize2`. RandomSize1 and RandomSize2 do not necessarily need to be set as multiples of k. We set them proportional to k just to keep the calculation simple. RS_2 has a subset of well-scattered data points from RS_1. This greatly reduces the dependence of the clustering result on the content of RS_1 and improves the effectiveness of the algorithm.

```
Algorithm COID (k: No. of Clusters)

Begin
    {1. Initialization Phase}
        repeat times = 0;
    Repeat
        Const1 and Const2 are two constants, Const1 > Const2;
        RandomSize1 ← Const1×k, RandomSize2 ← Const2×k;
        $RS_1$ ← random sample with the size of Randomsize1;
        $RS_2$ ← FindKMedoids($RS_1$, RandomSize2);
        {Assign data points to medoids to form medoid-associated sets}
        E ← DispatchDataPoints($RS_2$);
        {E is set of initial data division;}
        {Determine the characteristics of the medoid-associated sets}
        {$\mathcal{C}$ and $\mathcal{O}$} ← ClusterOrOutlier();
    Until($|\mathcal{C}| \geq$ k or repeat times $\geq \Im$)
        If $|\mathcal{C}| <$ k
          If there are less than Randomsize2 - $|\mathcal{C}|$ medoid-associated sets in E - $|\mathcal{C}|$
          with size $> 1$
            merge medoids-associated sets in E - $|\mathcal{C}|$ until there are k-$|\mathcal{C}|$ new sets
            with size $> 1$;
          Choose k-$|\mathcal{C}|$ largest medoid-associated sets in E with sizes $< T_s$ and
          move them to $\mathcal{C}$;

    {2. Iterative Phase}
    $\mathcal{C}$ ← MergeCluster($\mathcal{C}$);
    Repeat
    {Find the nearest cluster for each outlier}
        For each outlier o $\in \mathcal{O}$ do
        Begin
            Find its nearest cluster $\in \mathcal{C}$
        End
        Sort current set of clusters in ascending order based on their qualities;
        Sort current set of outliers in ascending order based on their qualities;
        {Reorganize the structure of clusters and outliers}
        ExchangeClusterAndOutlier();
        $\mathcal{O}'$ is the set of outliers with worst qualities;
        BDP is the set of boundary data points with worst qualities;
        U ← $\mathcal{O}' \cup$ BDP;
    Until(U is stable or iteration number $\geq \Im$)
End.
```

Fig. 3 Algorithm: COID

4.1.2 Dispatching data points

Once the smaller random set RS_2 of medoids is obtained, we proceed to determine which medoids are in some clusters and which medoids are actually outliers.

We first assign each data point dp to a certain medoid (the nearest one to dp) in RS_2. After this step, each $medoid_i \in RS_2$ is associated with a set of data points denoted as E_i in Fig. 4 (notice that E_i can be an empty set).

4.1.3 Initial partition of the data set

Cluster or outlier? After obtaining the set $E = \{E_i\}$ of the initial partition of the input data set $\mathbf{X}$, we check the size of each medoid-associated data subset $E_i \in E$. A medoid $medoid_i$

```
Proc DispatchDataPoints ($RS_2$: Set of Initial Medoids)
Begin
    RandomSize2 $\leftarrow |RS_2|$;
    For each $i = 1, \ldots, RandomSize2$ do
        $E_i \leftarrow \emptyset$
    For each data point $dp$ in the current data set do
    Begin
        Find $medoid_i \in RS_2$ with smallest $d(dp, medoid_i)$;
        $E_i = E_i \cup \{dp\}$
    End
    $E \leftarrow E_1, E_2, \ldots, E_{RandomSize2}$;
    Return $E$
End.
```

Fig. 4 Proc: DispatchDataPoints

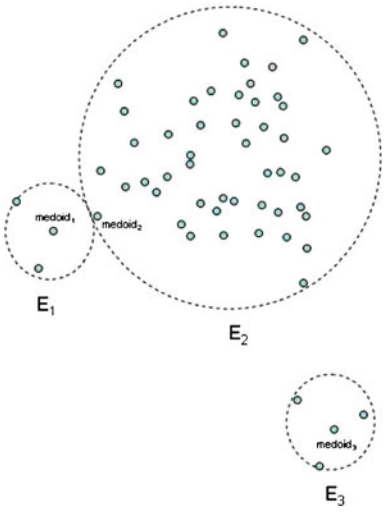

Fig. 5 Data groups associated with different medoids

with a small associated data subset E_i may seem to be an outlier, but it is not necessarily the case. Since we get the initial medoids randomly, multiple medoids can be located in the same natural cluster. Some of the medoids may contain very few data points, but they actually belong to a cluster instead of being an outlier. Figure 5 shows an example of such a case. In Fig. 5, three medoids ($medoid_1$, $medoid_2$, and $medoid_3$) are randomly selected, and they form three data subsets (E_1, E_2, and E_3). The data subset E_1 has the smallest size. However, it is easy to see that E_1 is actually a part of a natural cluster (combination of E_1 and E_2). Therefore, the size alone is not enough to determine whether a medoid is an outlier or actually in a cluster.

Here, we exploit an approach to adjust the criterion to determine whether a medoid is an outlier. We partition the data subsets $E_i \in E$ into sets $\mathcal{C}$ and $\mathcal{O}_{temp}$ by T_s. T_s is the threshold of the cluster size, which can be application oriented. For each subset E_s (formed by medoid $medoid_i$) in $\mathcal{O}_{\text{temp}}$, we use two factors to determine the possibility of $medoid_i$ being an outlier: the size of E_s and the minimum distance between E_s and other medoid-associated data subsets in E. The reason is given above. Practically, we use $\frac{|E_s|}{d_t(E_s)}$ as the criterion for the determination. $|E_s|$ is the size of E_s, and $d_t(E_s)$ is the diversity between E_s and the

```
Proc ClusterOrOutlier
    ($E$: set of initial data division, $T_s$: threshold of the cluster size)

Begin
  {$\mathcal{C}$ is the set of clusters, $\mathcal{O}$ is the set of outliers }
  {$\mathcal{O}_{temp}$ is the intermediate set of data subsets associated with medoids which might be outliers}
  $\mathcal{C} \leftarrow \emptyset$
  $\mathcal{O} \leftarrow \emptyset$
  $\mathcal{O}_{temp} \leftarrow \emptyset$
  {$E$ is the initial data partition }
  For each $E_i \in E$ do
  Begin
      {$T_s$ is the threshold of the cluster size}
      If $|E_i| \geq T_s$
          $\mathcal{C} = \mathcal{C} \cup \{E_i\}$
      Else
          $\mathcal{O}_{temp} = \mathcal{O}_{temp} \cup \{E_i\}$
      End
  For each $E_s \in \mathcal{O}_{temp}$ do
  Begin
      Find the nearest set $E_j \in E$ of $E_s$
      {$D_2$ is the diversity between two sets }
      $d_t(E_s) \leftarrow D_2(E_s, E_j)$
  End
  Sort $E_s \in \mathcal{O}_{temp}$ in ascending order by $\frac{|E_s|}{d_t(E_s)}$
  Denote the ordered set list as $l_s$
  Find the first sharp upward point $\tilde{p}$ in $l_s$
  $E\prime \leftarrow$ data subsets before $\tilde{p}$ in $l_s$
  $E\prime\prime \leftarrow$ data subsets after $\tilde{p}$ in $l_s$
  $\mathcal{O} \leftarrow$ data points $\in E\prime$
  $\mathcal{C} = \mathcal{C} \cup E\prime\prime$
  Return $\mathcal{C}$ and $\mathcal{O}$
End.
```

Fig. 6 Proc: ClusterOrOutlier

medoid-associated data subsets $E_j \in E$. See Fig. 6 for the definition of $d_t(E_s)$. The smaller the value of $|E_s|$ is and the larger $d_t(E_s)$ is, the smaller $\frac{|E_s|}{d_t(E_s)}$ is, and the medoid $medoid_i$ (forming data subset E_s) is probably an outlier. This is also consistent with the situation shown in Fig. 5.

For $E_s \in \mathcal{O}_{\text{temp}}$, we sort them in ascending order by $\frac{|E_s|}{d_t(E_s)}$ into an ordered list l_s and find the first sharp upward point $\tilde{p}$ (see Fig. 7). Data subsets before $\tilde{p}$ in l_s can be reasonably regarded as the groups of outliers since they have small size and also are far from other data subsets. When there is no sharp point at all in l_s, we simply set $\tilde{p}$ to the one-third position. Data subsets after $\tilde{p}$ in l_s are more likely some data groups inside natural clusters, and we put them back into the cluster set $\mathcal{C}$.

Real data sets such as those from UCI Machine Learning Repository [7] have data objects much greater than k. In most cases, $|\mathcal{C}|$ (the size of the cluster $\mathcal{C}$) is larger than k because of the way RS_2 is generated as described in 4.1.1. RS_2 has a subset of well-scattered data objects from RS_1. After the process of dispatching data points in 4.1.2, each medoid will

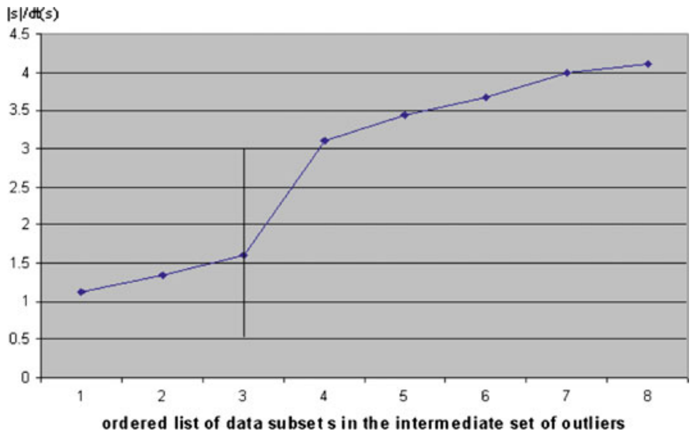

Fig. 7 Ordered list of outlier subsets in the intermediate set based on $\frac{|E_s|}{d_t(E_s)}$

associate a number of data objects. Together, they tend to form similar sized groups, since RS_2 better represents the distribution of the data set.

However, there is still a chance that $|\mathcal{C}|$ is less than k. In this case, to improve the result, we can adjust the threshold of the cluster size T_s. Improper value for T_s will cause the initialization phase to fail, as the initialization phase will always result in less than k initial clusters. For example, if T_s is set extremely high, such as $\lfloor n/2 \rfloor$, where n is the total number of data objects in the data set, we cannot obtain k clusters when k is larger than 1. Since RS_2 contains `RandomSize2` medoids, a more reasonable value for T_s is $\lfloor n/\texttt{RandomSize2} \rfloor$. In this way, the average number of the data objects associated with each medoid in RS_2 is in the same order of magnitude as T_s. So, the possibility that we have k or more initial clusters will be much higher. After several attempts, if we still cannot obtain at least k clusters, we then add k-$|\mathcal{C}|$ sets in E to $\mathcal{C}$. Those k-$|\mathcal{C}|$ sets are closest but not beyond the threshold T_s.

4.2 Iterative phase

In the iterative phase, there are four steps: (1) we merge the initial set of clusters into k clusters; (2) we sort clusters and outliers based on their qualities and select the worst clusters and outliers; (3) for clusters of the worst qualities, we exploit some methods to select the boundary data points for each of them; and (4) we refine the set of clusters and outliers gradually by exchanging the selected boundary data points and the worst outliers. Steps two, three, and four are performed iteratively until a certain termination condition is reached. We detail each step in the following.

4.2.1 Merging clusters

Before we perform the cluster–outlier iterative detection process, we should first merge the current cluster set $\mathcal{C}$ into k clusters. It is an iterative process. In each iteration, two nearest clusters are found in $\mathcal{C}$, and they are merged. The distance between two clusters C_1 and C_2 is based on the diversity measurement $D_2(C_1, C_2)$ of two clusters defined in Sect. 3. The iteration step is performed until the total number of clusters in $\mathcal{C}$ is k. We also compute the centroid of each cluster $C_i \in \mathcal{C}$ (denoted as c_i).

4.2.2 Sorting clusters and outliers

For each outlier $O \in \mathcal{O}$, its nearest cluster $C_o \in \mathcal{C}$ is located. The distance between the cluster C and the outlier O is based on the diversity measurement $D_1(C, O)$ defined in Sect. 3. The quality $Q_o(O)$ (defined in Sect. 3) is also calculated. Outliers with the worst qualities (similar to the method shown in Fig. 7, the outliers before the first sharp point) are put into set $\mathcal{O}'$. Similarly, for each cluster $C \in \mathcal{C}$, its quality $Q_c(C)$ (defined in Sect. 3) is calculated. Clusters with the worst qualities (similar to the method shown in Fig. 7, the clusters before the first sharp point) are put into set $\mathcal{C}'$.

If a cluster has bad quality, it is more apt to contain some data points better to be considered as outliers. Similarly, an outlier with bad quality should be considered to be included in a certain cluster.

4.2.3 Finding boundary data points

We define boundary data points (BDP) of a cluster as those with the farthest distance to the centroids of the cluster and having the least number of neighboring data points. Having the least number of neighboring data points ensures that our method does not only favor clusters of standard geometries such as hyper-spherical ones. For real-world data sets, the boundary data points of the clusters are ambiguous. Some boundary data points under one certain scale could be regarded as outliers under another scale. Some outliers under one certain scale may better to be treated as boundary data points under a different scale. To decide which data points should be boundary data points belonging to a certain cluster and which ones should be outliers is a difficult problem, and we design an algorithm to solve such a problem.

For each cluster $C_i \in \mathcal{C}'$, we identify its boundary data points based on d_{ij} and $\varsigma(j).d_{ij}$ is the distance between each data point $dp_j \in C_i$ and the centroid c_i of $C_i.\varsigma(j)$ is the number of data points within radius τ of dp_j, and τ is set proportional to $|C_i|$. We sort the data points $dp_j \in C_i$ in descending order by $\frac{d_{ij}}{\varsigma(j)}$ and locate the first sharp downward point $\tilde{p}$. Data points before $\tilde{p}$ are regarded as boundary data points of C_i. The set of boundary data points of C_i can be $\emptyset$ (empty) if there are no prominent boundary data points in C_i.

We then group all boundary data points of clusters in $\mathcal{C}'$ into a set BDP and check its size. If the size of BDP is much larger than the size of $\mathcal{O}'$, we should further reduce the size of BDP. Excessive differences between the sizes of BDP and $\mathcal{O}'$ could unbalance the initial division of $\mathbf{X}$ in the consideration that the initial results of most existing algorithms already contain clusters and outliers similar to the ground truth.

To reduce the size of BDP, we perform another sorting on the data points in BDP. Note that the previous sort of data points is for each cluster $C_i \in \mathcal{C}'$ individually while this time the sort is on the combination of the boundary data points from all the clusters in $\mathcal{C}'$. The criterion of the sort is also different from the previous one. The new criterion consists of (1) the distance between boundary data points and the centroid of clusters they belong to and (2) the number of their neighbors plus integrating the information of the sizes of the clusters. The rationale is because clusters in $\mathcal{C}'$ have difference sizes, and sort solely based on the distance measurement is unfair to the boundary data points of some clusters in this case. Practically, we use $\frac{d_{ij}}{|C_i| * \varsigma(j)}$ as the sorting criterion to eliminate the effect of the size differences among clusters. We reduce the size of BDP, making it only contain the first $|\mathcal{O}'|$ boundary data points in the ordered boundary data point list.

4.2.4 Exchanging data points in clusters and outliers

The next step is to exchange the characteristics of outliers and boundary data points. For each outlier $O \in \mathcal{O}'$, we add O into its nearest cluster. For each boundary data point $bdp \in$ BDP, we change it into a new outlier. The reason that we do not exchange boundary data points *between* two clusters is that the whole data partition will be deteriorated after exchange.
Termination condition: After the exchange step, we reorganize the united set U of the worst outlier set $\mathcal{O}'$ and the reduced boundary data point set BDP as $U = \mathcal{O}' \cup$ BDP. The iterative process is performed until either of the following two conditions is satisfied: the elements in U do not change dramatically anymore, or some threshold of the iteration number is reached.

4.3 Time and space analysis

Let the size of the data set be n. Throughout the process, we need to keep track of the information of all points, which collectively occupies $O(n)$ space. For the iteration step, we need (1) information of current set $\mathcal{C}$ of clusters, (2) the current set $\mathcal{O}$ of outliers, (3) the boundary data points of each cluster, and (4) the worst outliers and worst clusters in each iteration. The total space needed is $O(n)$. The time required for each iteration is $O(n + |\mathcal{C}| \log |\mathcal{C}| + |\mathcal{O}| \log |\mathcal{O}|)$ mainly for computation of the various qualities and sort process. The total time required for the algorithm is $O(\Im * (n + |\mathcal{C}| \log |\mathcal{C}| + |\mathcal{O}| \log |\mathcal{O}|))$ in which $\Im$ is the threshold of the iteration number.

5 Experiments

We conducted comprehensive experiments on both synthetic and real data sets to assess the accuracy and efficiency of the proposed approach. Our experiments were run on a SUN ULTRA 60 workstation with the Solaris 5.8 system. Trials using data sets from real-world applications were performed comparing to other algorithms such as CURE and Shrinking.

The accuracy of a detected cluster was measured by *precision* and *recall*. For a detected cluster C_i^s and a real cluster C_i^o, the *precision* of C_i^s with respect to C_i^o is defined as $\frac{|C_i^s \bigcap C_i^o|}{|C_i^s|}$ and the *recall* is $\frac{|C_i^s \bigcap C_i^o|}{|C_i^o|}$.$C_i^s$ is called a corresponding cluster of C_i^o if the precision and recall of C_i^s with respect to C_i^o are high.

5.1 Scalability on high-dimensional data sets

To test the scalability of our algorithm over dimensionality and data size, we designed a synthetic data generator to produce data sets with clusters. The sizes of the data sets varied from 5,000, 10,000, . . ., to 100,000, and the dimensions of the data sets varied from 10, 20, . . . to 60. Each data set contained five clusters with the points in each cluster generated in normal distributions. An additional 5% of data points were added randomly to each data set as noise. Clusters and outliers were effectively detected by our algorithm in all tests performed on these high-dimensional data sets. Figure 8 shows the running time of 20 groups of data sets with dimensions increasing from 10 to 60. Each group had a fixed data size (the noisy data points were not counted). Figure 9 shows the running time of 11 groups of data sets with sizes increasing from 5,000 to 100,000. Each of these 11 groups had a fixed number of dimensions. The two figures indicate that our algorithm is scalable over dimensionality and data size.

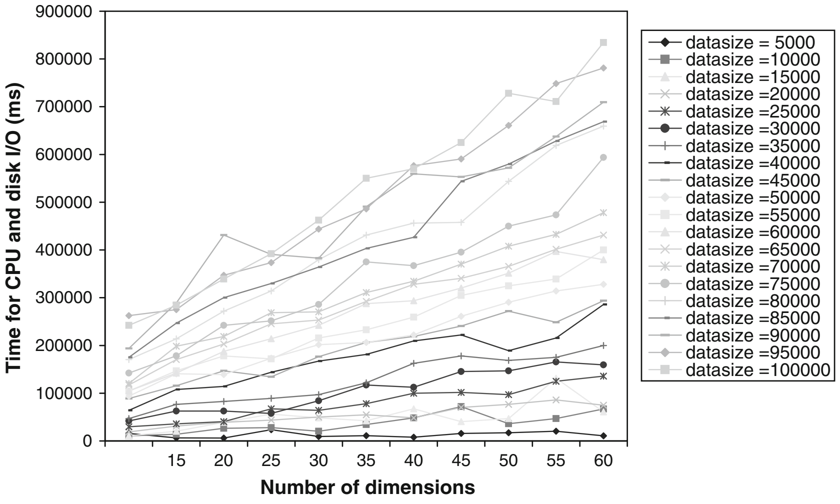

Fig. 8 Running time with increasing dimensions

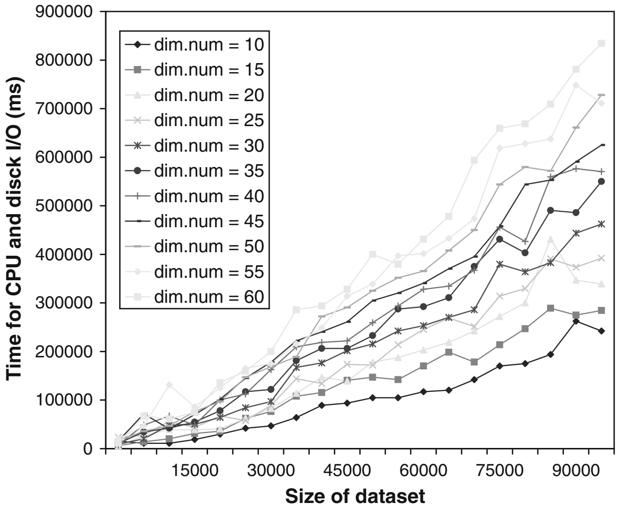

Fig. 9 Running time with increasing data sizes

5.2 Real-world data sets

The real data sets were obtained from UCI Machine Learning Repository [7]. The first set is the *Wine Recognition* data (simplified as Wine data). It contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It has 178 instances with 13 features. The data set has three clusters with the sizes of 59, 71, and 48.

The second data set is Ecoli data. It contains data regarding Protein Localization Sites. This data set is made up of 336 instances, with each instance having seven features. It has 8 clusters with the sizes of 143, 77, 52, 35, 20, 5, 2, and 2.

The third data set is *Pen-Based Recognition of Handwritten Digits* data (or Pendigits). It was created by collecting 250 samples from 44 writers. It has two subsets used, respectively, for training and testing. For the purpose of this experiment, we have combined these two subsets, resulting in a combined dataset with 10992 instances, each containing 16 attributes. The data set has ten clusters, C_i^o for $i = 1, 2, \ldots, 10$.

Due to the space limitation, we only show the comparison of the testing results of our algorithm on Wine and Ecoli data with those of other algorithms. We also demonstrate how COID can improve the performance of other algorithms on Wine and Pendigits data.

5.3 COID on wine data: An illustration

In this section, we used the Wine data set to illustrate how the COID algorithm performed. As mentioned in Sect. 4, the COID algorithm proceeded in two phases: an *initialization* phase and an *iterative* phase. We demonstrated how the original data points from the wine data were processed step by step, eventually forming various clusters and outliers.

5.3.1 Initialization Phase

1. ***Acquirement of medoids***

The first step in the initialization phase was to locate the initial set of medoids. We tried to find a *superset* of a good set of k medoids. From the ground truth, there were 3 ($k = 3$) natural clusters in the Wine data set. Here, we let `Const1` be 4 and `Const2` be 2 (Fig. 3). We also set the density threshold as 20, which means any group that contains less than 20 data points would first be regarded as a group of outliers (which could be changed later). The worst 3 clusters and the worst 3 outliers (in terms of the qualities) would be chosen in each iteration to perform exchange between outliers and boundary data points.

Based on these settings, we randomly chose 12 (`RandomSize1` = `Count1`$\times k$) data points from the Wine data set as the initial group `RS1` of medoids. Since those medoids were randomly picked, they might not represent the distribution of the data set well. We then applied the strategy described in Sect. 4.1.1 to find another superset `RS2` of the k ($k = 3$) medoids with size 6 (`RandomSize2` = `Count2`$\times k$). We started with a random pick of a medoid from `RS1` and then performed an iteration to pick up the next medoid, until we chose 6 medoids. Each time, we tried to find the medoid that was farthest to the medoids that were already in `RS2`.

Now, we had 6 medoids in RS2. We then assigned each data point in the Wine data to one of these 6 medoids. There were 178 data points altogether. After the assignment, we had initially 6 groups of data points represented by these 6 medoids. The sizes were 21, 42, 45, 28, 18, and 24.

2. ***Dispatching data points***

The next step was to decide which groups were initial clusters and which ones were initial outliers. Since the density threshold for a cluster was set as 20, at this stage, we had 5 clusters and 1 group of outliers (the one with size 18). If there were more than 10 groups of outliers, we would sort them in ascending order based on $\frac{|E_s|}{d_t(E_s)}$ (Fig. 6) and found the first sharp upward point $\tilde{p}$ (Fig. 7). Since here we only had 1 group of outliers, this step could be skipped. The group of 18 data points would be assigned as the initial outliers. So far, we partitioned the Wine data set into 5 clusters and 18 outliers.

5.3.2 Iterative phase

1. ***Merging clusters***

We now entered iterative phase. Since there were 3 ($k = 3$) clusters in the Wine data set according to the ground truth, the first step in the iterative phase was to merge clusters eventually to form 3 clusters. Each time we found two clusters that were closest to each other and merged them together. We first merged the cluster with 21 data points and the one with 24 data points. The new cluster had 45 data points. We then merged the cluster with 42 data points and the one with 28 data points. The new cluster had 70 data points. After merging clusters twice, now we shrunk the number of clusters from 5 to 3, which was the number of natural clusters according to the ground truth.

2. Iteration process

We performed the following steps iteratively: (1) for each cluster, we found its boundary data points following the strategy described in Sect. 4.2.3; (2) for each outlier, we found the nearest cluster it was close to; (3) we sorted outliers and clusters respectively based on their qualities and selected the worst outliers and the worst boundary data points; and (4) we changed the worst outliers to the new boundary data points, and we changed the worst boundary data points to the new outliers.

We assigned each data point in the Wine data set with an identity number (ID), from $0, 1, \ldots, 177$. We assigned clusters as Cluster 1 $\left(C_1^s\right)$ contained 70 data points; Cluster 2 $\left(C_2^s\right)$ contained 45 data points; Cluster 3 $\left(C_3^s\right)$ contained 45 data points. We used C_1^o, C_2^o, and C_3^o to denote the natural clusters according to the ground truth of the Wine data set. Before iteration started, C_1^s contained data points from C_1^o, C_2^o, and C_3^o. Its *precision* and *recall* were 0.785 and 0.932, respectively. The detail for each cluster is listed below.

Cluster	Size	From C_1^o	From C_2^o	From C_3^o	Precision	Recall
C_1^s	70	55	13	2	0.785	0.932
C_2^s	45	4	40	1	0.888	0.563
C_3^s	45	0	3	42	0.933	0.875

In iteration 1, C_1^s had 2 boundary data points, C_2^s had 2 boundary data points, and C_3^s had 2 boundary data points. The worst BDP IDs were 33, **90**, 121, 146, and 169. They were changed as new outliers. There were 5 worst outliers, and their IDs were 75, 117, 100, 62, and 86. For each O of those 5 outliers, we found the cluster C it is closest to based on $D_1(C, O)$ defined in Sect. 3 and included it in C.

In iteration 2, the worst BDP IDs were 7, 61, and 158, and they were changed as new outliers. There were 3 worst outliers, and their IDs were **90**, 64, and 67. These outliers were included in various clusters as we did in iteration 1. There was 1 data point with ID 90, which also appeared in the last iteration. After iteration two, clusters were updated as follows:

Cluster	Size	Boundary point	From C_1^o	From C_2^o	From C_3^o	Precision	Recall
C_1^s	69	2	55	13	1	0.797	0.932
C_2^s	48	2	3	44	1	0.916	0.619
C_3^s	43	2	0	2	41	0.953	0.854

In iteration 3, the worst BDP IDs were 90, 145, 95, and 68. They were changed as new outliers. There were 4 worst outliers, and their IDs were 61, 142, 96, 112. They were changed as new data points for certain clusters. There were 2 data points with ID **90** and **61**, which also appeared in the last iteration. After iteration two, clusters were updated as follows:

Cluster	Size	Boundary point	From C_1^o	From C_2^o	From C_3^o	Precision	Recall
C_1^s	68	1	55	13	0	0.808	0.932
C_2^s	49	2	2	46	1	0.938	0.647
C_3^s	43	1	0	2	41	0.953	0.854

The iterations were preformed until the union of worst DBPs and worst outliers was stable or the iteration number exceeded a threshold. In this case, we executed 30 times, and the union of worst DBPs and worst outliers became stable.

Cluster	Size	Boundary point	From C_1^o	From C_2^o	From C_3^o	Precision	Recall
C_1^s	68	1	57	9	0	0.876	0.966
C_2^s	60	2	2	57	1	0.950	0.802
C_3^s	43	1	0	2	44	0.956	0.916

5.4 Algorithm comparison

5.4.1 Algorithms

CURE: We used the implementation of CURE provided to us by Michael Steinbach from University of Minnesota. It requires three input parameter options: $-k$ option is for the number of clusters, $-\alpha$ is for alpha parameter of CURE, and $-r$ is the number of representative points of the cluster. To compare CURE with our algorithm fairly, we applied different values of those parameters extensively and adopted the best clustering results. Since we used CURE mainly to compare the accuracy of its clustering result with ours, we did not take into consideration the partition number parameter p for speedup mentioned in [17].

Shrinking: We also conducted experiments using Shrinking algorithm [36] on real data sets. Shrinking is a data preprocessing technique that optimizes the inner structure of data inspired by the Newton's Universal Law of Gravitation [31] in the real world. Shrinking-based multi-dimensional data analysis approach first moves data points along the direction of the density gradient, thus generating condensed, widely separated clusters. It then detects clusters by finding the connected components of dense cells. Then, it uses a cluster-wise evaluation measurement to compare the clusters from different cases and select the best clustering results.

COID: Our clustering and outlier detection version is based on the algorithm described in the previous sections. First, medoids for natural clusters are searched and the initial clusters and outliers are detected. Then, clusters and outliers are iteratively modified and refined according to the relationship among them.

5.4.2 Comparison

The first experiments were conducted on Wine data. We applied CURE algorithm on the Wine Recognition data set, setting parameter values to different values extensively. We set the cluster number parameter k to 3 based on the ground truth of the Wine Recognition data set, set the shrinking factor α and the number of representative points r with different values

Table 1 Clustering result of CURE for Wine data as $\alpha = 0.3$ and $r = 10$

	C_1^s	C_2^s	C_3^s	Recall (%)
C_1^o	54	3	2	91.52
C_2^o	2	41	9	57.77
C_3^o	10	5	26	54.16
Precision (%)	75.00	82.00	56.52	

Table 2 Clustering results of Shrinking algorithm for Wine data

	C_1^s	C_2^s	C_3^s	Recall (%)
C_1^o	53	1	1	89.83
C_2^o	0	51	1	71.83
C_3^o	0	0	43	89.58
Precision (%)	100	98.08	93.48	

Table 3 Clustering result 1 of COID algorithm for Wine data

	C_1^s	C_2^s	C_3^s	Recall (%)
C_1^o	53	3	1	89.83
C_2^o	1	65	2	91.54
C_3^o	2	2	45	93.75
Precision (%)	94.64	89.04	97.82	

Table 4 Clustering result 2 of COID algorithm for Wine data

	C_1^s	C_2^s	C_3^s	Recall (%)
C_1^o	53	2	2	89.83
C_2^o	1	64	3	90.14
C_3^o	0	1	45	93.75
Precision (%)	96.36	86.49	97.82	

and found the best clustering result among those parameter values. Due to space limitation, we just present one of the best results of CURE. Table 1 shows the clustering results of CURE algorithm when $\alpha = 0.3$, and $r = 10$.

We then applied Shrinking-based clustering algorithm on Wine data. Table 2 shows the clustering results of the Shrinking algorithm.

Our COID algorithm was also performed on the Wine data. In order to analyze how randomly generated medoids in RS_1 can affect the performance of our approach, we randomly generated medoids three times. Tables 3, 4, and 5 show the clustering results of our algorithm. The overall clustering result of COID algorithm was better than CURE (in Table 1). As for Shrinking, although the average precision of the testing result of COID was lower than that of Shrinking (93.50 vs. 97.05%), the average recall of the testing result of COID was higher than that of Shrinking (91.36 vs. 83.75%).

Experiments were performed on Ecoli data. We applied CURE algorithm on the Ecoli data set, setting parameter values to different values extensively. According to the ground truth of the Ecoli data set, there are 8 clusters in it. However, three of the clusters are too small, so we set the cluster number parameter k to 5, set the shrinking factor α and the number of

Table 5 Clustering result 3 of COID algorithm for Wine data

	C_1^s	C_2^s	C_3^s	Recall (%)
C_1^o	53	2	2	88.14
C_2^o	1	64	3	91.54
C_3^o	0	1	45	93.75
Precision (%)	94.54	89.04	95.74	

Table 6 Clustering result of CURE for Ecoli data as $\alpha = 0.8$ and r = 20

	C_1^s	C_2^s	C_3^s	C_4^s	C_5^s	C_6^s	C_7^s	C_8^s	Recall (%)
C_1^o	115	10	0	N/A	N/A	0	N/A	N/A	80.41
C_2^o	1	41	1	N/A	N/A	0	N/A	N/A	53.24
C_3^o	0	8	30	N/A	N/A	0	N/A	N/A	57.69
C_4^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C_5^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C_6^o	1	0	0	N/A	N/A	3	N/A	N/A	60.00
C_7^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C_8^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Precision (%)	95.83	61.19	93.75	N/A	N/A	100	N/A	N/A	

Table 7 Clustering result of Shrinking algorithm for Ecoli data

	C_1^s	C_2^s	C_3^s	C_4^s	C_5^s	C_6^s	C_7^s	C_8^s	Recall (%)
C_1^o	130	3	5	1	0	N/A	N/A	N/A	90.91
C_2^o	2	22	14	7	1	N/A	N/A	N/A	28.57
C_3^o	0	0	43	5	0	N/A	N/A	N/A	82.69
C_4^o	0	0	2	32	0	N/A	N/A	N/A	91.43
C_5^o	0	0	3	2	10	N/A	N/A	N/A	50.00
C_6^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C_7^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C_8^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Precision (%)	96.30	100	63.24	65.31	90.91	N/A	N/A	N/A	

representative points r with various of values and select the best clustering result. Table 6 shows the clustering results of CURE algorithm when α is equal to 0.8 and r is 20.

We then applied Shrinking-based clustering algorithm on Ecoli data. Table 7 shows the clustering results of the Shrinking algorithm.

Our COID algorithm was also performed for three times on the Ecoli data, generating three different sets of medoids for RS_1. Table 8, 9, and 10 show the clustering results of our algorithm. Compared to the clustering results of CURE and Shrinking algorithms, COID has the most information of the first 3 largest natural clusters.

Table 8 Clustering result 1 of COID algorithm for Ecoli data

	C^s_1	C^s_2	C^s_3	C^s_4	C^s_5	C^s_6	C^s_7	C^s_8	Recall (%)
C^o_1	130	2	4	N/A	N/A	0	N/A	N/A	90.90
C^o_2	3	56	5	N/A	N/A	0	N/A	N/A	72.72
C^o_3	1	4	47	N/A	N/A	0	N/A	N/A	90.38
C^o_4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_6	0	0	0	N/A	N/A	4	N/A	N/A	80.00
C^o_7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Precision (%)	94.89	62.92	67.14	N/A	N/A	100	N/A	N/A	

Table 9 Clustering result 2 of COID algorithm for Ecoli data

	C^s_1	C^s_2	C^s_3	C^s_4	C^s_5	C^s_6	C^s_7	C^s_8	Recall (%)
C^o_1	131	3	3	N/A	N/A	0	N/A	N/A	91.60
C^o_2	4	56	3	N/A	N/A	0	N/A	N/A	72.72
C^o_3	1	3	47	N/A	N/A	0	N/A	N/A	90.38
C^o_4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_6	0	0	0	N/A	N/A	4	N/A	N/A	80.00
C^o_7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Precision (%)	94.92	62.22	68.12	N/A	N/A	100	N/A	N/A	

Table 10 Clustering result 3 of COID algorithm for Ecoli data

	C^s_1	C^s_2	C^s_3	C^s_4	C^s_5	C^s_6	C^s_7	C^s_8	Recall (%)
C^o_1	130	2	3	N/A	N/A	0	N/A	N/A	90.90
C^o_2	3	55	6	N/A	N/A	0	N/A	N/A	71.43
C^o_3	2	3	47	N/A	N/A	0	N/A	N/A	90.38
C^o_4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_6	0	0	0	N/A	N/A	4	N/A	N/A	80.00
C^o_7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C^o_8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Precision (%)	94.20	61.80	66.20	N/A	N/A	100	N/A	N/A	

5.5 Using COID to improve other algorithms

We also conducted experiments to test how COID algorithm can improve the clustering and outlier detection results of other algorithms. Due to the space limitation, here we only

Table 11 Iteration of COID algorithm on clustering result of CURE algorithm on Wine data

i	$\mathcal{O}'$	BDP	Match	Average precision (%)	Average Recall (%)
0	N/A	N/A	N/A	71.17	67.81
1	70 36	110 66	0	71.24	67.90
2	44 19 74	98 109 124	0	71.91	68.56
...	...	...	...	...	...
20	129 62	124 59	3	82.96	79.32
21	**124 110**	**62 43**	2	83.44	79.79
22	**62 43**	**110 124**	4	83.25	79.42

demonstrate how COID can improve the testing results of Wine data and Pendigits data from the CURE algorithm.

Table 11 shows the change in status of the testing result of Wine data. In Table 11, the column $\mathcal{O}'$ contains ids of those outliers with the worst qualities, although there might be other outliers at that moment. Similarly, the column BDP presents ids of those boundary data points of clusters that had worst qualities. The contents of these two columns were to be exchanged with each other. They constituted the set U ($U = \mathcal{O}' \cup$ BDP). The column *match* shows how many elements in U in current iteration were similar to those in the previous iteration. The average precision and the average recall columns are according to the clustering results.

The first line lists the original testing result from the CURE algorithm on Wine data. So the worst outliers, worst boundary data points and the match were not available yet. The average precision was 71.17% (resulting from the average of the precisions of the clustering result from Table 1: 75.00, 82.00, and 56.52%). The average recall was 67.81% (resulting from the average of the recalls of the clustering result from Table 1: 91.52, 57.77, and 54.16%).

In each iteration, the clusters and outliers were adjusted and modified according to their intra-relationship and inter-relationship. Outliers with the worst qualities were selected, so were those worst boundary data points. From Table 11, we can see that the average precision and average recall of the clustering result improved as the iteration went on. The iterations were executed till iteration 22, when the match of the set U to that of the previous iteration 21 was 100%. The updated average precision was 83.25%, and the updated average recall was 79.42%. Both were much higher than the original ones.

We tested how COID could improve the performance of the CURE algorithm on Pendigits data as well. Table 12 shows the change in status of the testing result of Pendigits data.

6 Conclusion and discussion

In this paper, we present COID, a novel cluster and outlier iterative detection approach. The method can effectively and efficiently improve the qualities of clusters and outliers in a multi-dimensional noisy data sets. Clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached.

The COID approach can be applied in various fields such as networking security, financial analysis, credit card fraud detection, etc. In these fields, the pattern of the dense data groups

Table 12 Iteration of COID algorithm on clustering result of CURE algorithm on Pendigits data

i	$\mathcal{O}'$	BDP	Match	Average precision (%)	Average Recall (%)
0	N/A	N/A	N/A	79.22	49.69
1	3151 3994	3978 499	0	79.71	49.70
2	34 4040 2887	3994 3151 3318	2	79.71	49.70
3	3151 3994	2887 6838	3	78.78	49.69
...	...	...	...	...	...
23	3686 1216 824 10111 10718 1171 6493 7256 247 4336	1482 10986 10492 8047 499 8387 3686 8891 8643	8	81.58	49.83
24	**10111 10718 247 1482** 1171 6493 7256 4336	**10492 8387 3686 8891** **8643 499** 10986 8047	8	81.93	49.82
25	**10492 3686 499** **8891 8643 8387**	**247 1482 10111** **10718** 6838 4042	10	82.33	49.83

(clusters) can be arbitrary and sometimes vague, as well as the isolated data (outliers). Our approach can be applied to refine the clusters and outliers iteratively.

The cluster and outlier iterative detection approach still poses open issues. In some cases, clusters and outliers cannot be efficiently and effectively detected in full data space, and there might exist obstacles. Our future work includes exploring relationship between clusters and outliers in subspace data mining problems and studying cluster and outlier detection approaches in the dimension reduction field in the presence of obstacles.

References

1. Achtert E, Kriegel H, Zimek A (2008) ELKI: a software system for evaluation of subspace clustering algorithms. In: Ludascher B, Mamoulis N (eds) Proceedings of the 20th international conference on scientific and statistical database management (SSDBM), Hong Kong, pp 580–585
2. Aggarwal C, Yu P (2001) Outlier detection for high dimensional data. In: Aref W (ed) Proceedings of the 2001 ACM SIGMOD international conference on management of data. ACM Press, Santa Barbara, pp 37–46
3. Agrawal R, Gehrke J, Gunopulos D et al (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Haas L, Tiwary A (eds) Proceedings of the ACM SIGMOD international conference on management of data. ACM Press, Seattle, pp 94–105
4. Aggarwal C, Hinneburg A, Keim D (2001) On the surprising behavior of distance metrics in high dimensional space. In: Bussche J, VianuLecture V (eds) Proceedings of the 8th international conference on database theory. Springer, London, pp 420–434
5. Aggarwal C, Procopiuc C, Wolf J et al (1999) Fast algorithms for projected clustering. In: Delis A, Faloutsos C, Ghandeharizadeh S (eds) Proceedings of the ACM SIGMOD conference on management of data. ACM Press, Philadelphia, pp 61–72
6. Ankerst M, Breunig M, Kriegel H et al (1999) OPTICS: ordering points to identify the clustering structure. In: Delis A, Faloutsos C, Ghandeharizadeh S (eds) Proceedings of the ACM SIGMOD conference on management of data. ACM Press, Philadelphia, pp 49–60
7. Bay S (1999) The UCI KDD Archive [http://kdd.ics.uci.edu]. Department of Information and Computer Science, University of California, Irvine
8. Beyer K, Goldstein J, Ramakrishnan R et al (1999) When is "nearest neighbor" meaningful?. In: Beeri C, Buneman P (eds) Proceedings of international conference on database theory. Springer, Jerusalem, pp 217–235
9. Bradley P, Fayyad U (1998) Refining initial points for K-Means clustering. In: Proceedings of 15th international conference on machine learning. Morgan Kaufmann, San Francisco, pp 91–99

10. Breunig M, Kriegel H, Ng R et al (2000) LOF: identifying density-based local outliers. In: Chen W, Naughton J, Bernstein P (eds) Proceedings of the ACM SIGMOD conference on management of data. ACM, Dallas, pp 93–104
11. Chen Y, Tu L (2007) Density-based clustering for real-time stream data. In: Berkhin P, Caruana R, Wu X (eds) Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, San Jose, pp 133–142
12. Chen C, Lee J (2001) The validity measurement of fuzzy C-means classifier for remotely sensed images. In: Proceedings of 22nd Asian conference on remote sensing. Singapore
13. Ester M, Kriegel H, Sander J et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han J, Fayyad U (eds) Proceedings of 2nd international conference on knowledge discovery and data mining. AAAI Press, Portland, pp 226–231
14. Fayyad U, Piatetsky-Shapiro G, Smyth P et al (1996) Advances in knowledge discovery and data mining. AAAI Press, Menlo Park
15. Fayyad U, Reina C, Bradley P (1998) Initialization of iterative refinement clustering algorithms. In: Agrawal R, Stolorz P, Piatetsky-Shapiro G (eds) Proceedings of the fourth international conference on knowledge discovery and data mining. AAAI Press, New York, pp 194–198
16. Gonzalez T (1985) Clustering to minimize the maximum intercluster distance. Theor Comput Sci 38:311–322
17. Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for large databases. In: Haas L, Tiwary A (eds) Proceedings of the ACM SIGMOD international conference on management of data. ACM Press, Seattle, pp 73–84
18. Guha S, Rastogi R, Shim K (1999) ROCK: a robust clustering algorithm for categorical attributes. In: Proceedings of the IEEE conference on data engineering. IEEE Computer Society Press, Sydney, pp 512–521
19. Hinneburg A, Keim D (1998) An efficient approach to clustering in large multimedia databases with noise. In: Agrawal R, Stolorz P, Piatetsky-Shapiro G (eds) Proceedings of the fourth international conference on knowledge discovery and data mining. AAAI Press, New York, pp 58–65
20. Halkidi M, Vazirgiannis M (2001) A data set oriented approach for clustering algorithm selection. In: Raedt L, Siebes A (eds) Proceedings of the 5th European conference on principles of data mining and knowledge discovery. Springer, Freiburg, pp 165–179
21. Hinneburg A, Aggarwal C, Keim D (2000) What is the nearest neighbor in high dimensional spaces? In: Abbadi A, Brodie M, Chakravarthy S (eds) Proceedings of 26th international conference on very large data bases. Morgan Kaufmann, Cairo, pp 506–515
22. Jain A, Murty M, Flyn P (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
23. Karypis G, Han E, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. Computer 32:68–75
24. Kaufman L, Rousseeuw P (1990) Finding groups in data: an introduction to cluster analysis. Wiley, Hoboken
25. Knorr E, Ng R (1998) Algorithms for mining distance-based outliers in large datasets. In: Gupta A, Shmueli O, Widom J (eds) Proceedings of 24th international conference on very large data bases. Morgan Kaufmann, New York, pp 392–403
26. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. University of California Press, Berleley 1:281–297
27. Ng R, Han J (1994) Efficient and effective clustering methods for spatial data mining. In: Bocca J, Jarke M, Zaniolo C (eds) Proceedings of the 20th international conference on very large data bases. Morgan Kaufmann, Santiago de Chile, pp 144–155
28. Nguyen M, Mark L, Omiecinski E (2008) Unusual pattern detection in high dimensions. Advances in knowledge discovery and data mining, 12th Pacific-Asia conference. Springer, Osaka, pp 247–259
29. Peterson G, McBride B (2008) The importance of generalizability for anomaly detection. Knowl Inf Syst 14(3):377–392
30. Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: Chen W, Naughton J, Bernstein P (eds) Proceedings of the ACM SIGMOD conference on management of data. ACM, Dallas, pp 427–438
31. Rothman M (1963) The laws of physics. Basic Books, New York
32. Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: Gupta A, Shmueli O, Widom J (eds) Proceedings of 24th international conference on very large data bases. Morgan Kaufmann, New York, pp 428–439
33. Shi Y (2008a) Detecting clusters and outliers for multi-dimensional data. In: Proceedings of the 2008 international conference on multimedia and ubiquitous engineering. SERSC, Busan, pp 429–432

34. Shi Y (2008b) SubCOID: exploring cluster-outlier iterative detection approach to multi-dimensional data analysis in subspace. In: ACMSE 2008: the 46th ACM southeast conference. ACM, Auburn, pp 132–135
35. Shi Y, Zhang A (2005) Towards exploring interactive relationship between clusters and outliers in multi-dimensional data analysis. In: Proceedings of the 21st international conference on data engineering. IEEE Computer Society, Tokyo, pp 518–519
36. Shi Y, Song Y, Zhang A (2003) A shrinking-based approach for multi-dimensional data analysis. In: Freytag J, Lockemann P, Abiteboul S et al (eds) Proceedings of 29th international conference on very large data bases. ACM, Berlin, pp 440–451
37. Tao Y, Xiao X, Zhou S (2006) Mining distance-based outliers from large databases in any metric space. In: Eliassi-Rad T, Ungar L, Craven M et al (eds) Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Philadelphia, pp 394–403
38. Wang J, Chiang J (2008) A cluster validity measure with outlier detection for support vector clustering. IEEE Trans Syst, Man, Cybernet, B 38(1):78–89
39. Wang W, Yang J, Muntz R (1997) STING: a statistical information grid approach to spatial data mining. In: Jarke M, Carey M, Dittrich K et al (eds) Proceedings of 23rd international conference on very large data bases. Morgan Kaufmann, Athens, pp 186–195
40. Wu M, Jermaine C (2006) Outlier detection by sampling with accuracy guarantees. In: Eliassi-Rad T, Ungar L, Craven M et al (eds) Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Philadelphia, pp 767–772
41. Wu X, Kumar V, Ross Q et al (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
42. Xiong H, Steinbach M, Ruslim A et al (2008) Characterizing pattern preserving clustering. Knowl Inf Syst 19(3):311–336
43. Xiong H, Wu J, Chen J et al (2006) K-means clustering versus validation measures: a data distribution perspective. In: Eliassi-Rad T, Ungar L, Craven M et al (eds) Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Philadelphia, pp 779–784
44. Yang J, Zhong N, Yao Y et al (2008) Local peculiarity factor and its application in outlier detection. In: Li Y, Liu B, Sarawagi S (eds) Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Las Vegas, pp 776–784
45. Yu D, Sheikholeslami G, Zhang A (2000) FindOut: finding outliers in very large Datasets. Knowl Inf Syst 4(4):387–412
46. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Jagadish H, Mumick I (eds) Proceedings of the 1996 ACM SIGMOD international conference on management of data. ACM, Montreal, pp 103–114

Author Biographies

Yong Shi received the BS and MS degrees, both in computer science, from the University of Science and Technology of China in 1996 and 1999, respectively. He received Ph.D. in computer science from the State University of New York at Buffalo in 2006. Currently, he is an assistant professor in the Department of Computer Science and Information Systems in Kennesaw State University. His research interests include data mining, database, machine learning, and information retrieval.

Li Zhang received his BS and MS degrees in computer science in 1991 and 1998, MA in mathematics in 1997, and Ph.D. in computer science from the State University of New York at Buffalo in 2004. Currently, he is an associate professor of the department of computer science, Eastern Michigan University. His research interests include bioinformatics, data mining, pattern recognition, visualization, computer graphics, non-well-founded set theory, computational linguistic, knowledge representation, logical reasoning system, artificial intelligence, database and web services.