CURLER: Finding and Visualizing Nonlinear Correlation Clusters [*]

Anthony K. H. Tung   Xin Xu   Beng Chin Ooi

School of Computing
National University of Singapore
{atung,xuxin,ooibc}@comp.nus.edu.sg

ABSTRACT

While much work has been done in finding linear correlation among subsets of features in high-dimensional data, work on detecting nonlinear correlation has been left largely untouched. In this paper, we present an algorithm for finding and visualizing nonlinear correlation clusters in the subspace of high-dimensional databases.

Unlike the detection of linear correlation in which clusters are of unique orientations, finding nonlinear correlation clusters of varying orientations requires merging clusters of possibly very different orientations. Combined with the fact that spatial proximity must be judged based on a subset of features that are not originally known, deciding which clusters to be merged during the clustering process becomes a challenge. To avoid this problem, we propose a novel concept called *co-sharing level* which captures both spatial proximity and cluster orientation when judging similarity between clusters. Based on this concept, we develop an algorithm which not only detects nonlinear correlation clusters but also provides a way to visualize them. Experiments on both synthetic and real-life datasets are done to show the effectiveness of our method.

1. INTRODUCTION

In recent years, large amounts of high-dimensional data, such as images, handwriting and gene expression profiles have been generated. Analyzing and handling such kinds of data have become an issue of keen interest. Elucidating the patterns hidden in high-dimensional data imposes an even greater challenge on cluster analysis.

Data objects of high dimensionality are NOT globally correlated in all the features because of the inherent sparsity of the data. Instead, a cluster of data objects may be strongly correlated only in a subset of features. Furthermore, the na-

---

[*]Dedicated to the late Prof. Hongjun Lu, our mentor, colleague and friend who will always be remembered.
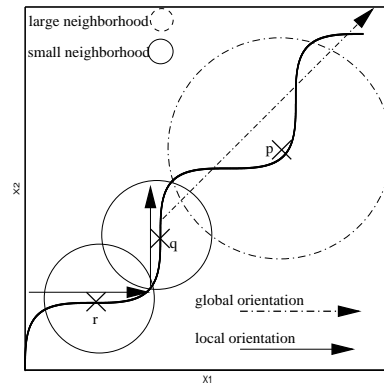
Figure 1: Global vs Local Orientation

ture of such correlation is usually local to a subset of the data objects, and it is possible for another subset of the objects to be correlated in a different subset of features. Traditional methods of detecting correlations like GDR [20] and PCA [16] are not applicable in this case since they can detect only global correlations in whole databases.

To handle the above problem, several subspace clustering algorithms such as ORCLUS [3] and 4C [7] have been proposed to identify local correlation clusters with arbitrary orientations, assuming each cluster has a fixed orientation. They identify clusters of data objects which are linearly correlated in some subset of the features.

In real-life datasets, correlation between features could however be nonlinear, depending on how the dimensions are normalized and scaled [14]. For example, physical studies have shown that the pressure, volume and temperature of an ideal gas exhibit nonlinear relationships. In biology, it is also known that the co-expression patterns of genes in a gene network can be nonlinear [12]. Without any detailed domain knowledge of a dataset, it is difficult to scale and normalize the dataset such that all nonlinear relationships become linear. It is even possible that the scaling and normalization themselves cause linear relationships to become nonlinear in some subset of the features.

Detecting nonlinear correlation clusters is challenging because the clusters can have both **local** and **global** orientations, depending on the size of the neighborhood being considered. As an example, consider Figure 1, which shows a 2D sinusoidal curve oriented at 45 degrees to the two axes.

Assuming the objects cluster around the curve, we will be able to detect the global orientation of this cluster if we consider a large neighborhood which is represented by the large circle centered at point p. However, if we take a smaller neighborhood at point q, we will only find the local orientation which can be very different from the global one. Furthermore, the local orientations of two points that are spatially close may not be similar at the same time, as can be seen from the small neighborhoods around q and r.

We next look at how the presence of local and global orientations may pose problems for existing correlation clustering algorithms like ORCLUS [3] and 4C [7]. These algorithms usually work in two steps. First, small clusters called **microclusters** [21, 22] are formed by grouping small number of objects that are near each other. Second, microclusters that are close both in proximity and orientation are merged in a bottom-up fashion to form bigger clusters. With nonlinear correlation clusters, such approaches will encounter two problems:

1) Determination of Neighborhood
Given that the orientation of a microcluster is sensitive to the size of the neighborhood from which its members are drawn, it is difficult to determine a neighborhood size in advance such that both the local and global orientations of the clusters are captured. Combined with the fact that spatial proximity must be judged based on a subset of the features that are not originally known, forming microclusters that capture the orientation of their neighborhood becomes a major challenge.

2) Judging Similarity between Microclusters
Since the orientations of two microclusters in close proximity can be very different, judging the similarity between two microclusters becomes non-trivial. Given a pair of microclusters which have high proximity [1] but very different orientations and another pair with similar orientations but low proximity, the order of merging for these two pairs cannot be easily determined. This in turn affects the final clustering result. One way to avoid this problem is to assign different weights to the importance of proximity and orientations, and then compute a combined similarity measure. However, it is not guaranteed that there will always be a unique weight assignment that gives a good global clustering result.

In this paper, we aim to overcome the above problems in finding nonlinear correlation clusters. Our contributions are as follows:

1. We highlight the existence of local and global orientations in nonlinear correlation clusters and explain how they pose problems for existing subspace clustering algorithms like ORCLUS [3] and 4C [7], which are designed to find linear correlation clusters.

2. We design an algorithm called CURLER [2], for finding and visualizing complex nonlinear correlation clusters. Unlike many existing algorithms which use a bottom-up approach, CURLER adopts an interactive

top-down approach for finding nonlinear correlation clusters so that both global and local orientations can be detected. A fuzzy clustering algorithm based on Expectation Maximization (EM) [15] is adopted to form the microclusters so that neighborhoods can be determined naturally and correctly. The algorithm also provides a similarity measure called **co-sharing level** that avoids the need to judge the importance of proximity and orientation when merging microclusters.

3. We present extensive experiments to show the efficiency and effectiveness of CURLER.

The rest of the paper is organized as follows. Related work is reviewed and discussed in Section 2. We formally present our algorithm in detail in Section 3. We discuss our experimental analysis in Section 4. We conclude in Section 5.

2. RELATED WORK
Clustering algorithms can be grouped into two large categories: full space clustering, to which most traditional clustering algorithms belong, and subspace clustering.

The clustering strategies utilized by full space clustering algorithms mainly include *partitioning based clustering*, which favors spherical clusters such as the k-medoid [17] family and EM algorithms [15]; and *density-based clustering*, represented by DBSCAN [11], DBCLASD [23], DENCLUE [2] and the more recent OPTICS [5]. EM clustering algorithms such as [19] compute probabilities of cluster memberships for each data object according to certain probability distribution; the aim is to maximize the overall probability of the data. For density-based algorithms, OPTICS is the algorithm most related to our work. OPTICS creates an augmented ordering of the database, thereby representing the density-based clustering structure based on 'core-distance' and 'reachability-distance'. However, OPTICS has little concern for the subspace where clusters exist or the correlation among a subset of features.

As large amounts of high-dimensional data have resulted from various application domains, researchers argue that it is more meaningful to find clusters in a subset of the features. Several algorithms for subspace clustering have been proposed in recent years.

Some subspace clustering algorithms like CLIQUE [4], OptiGrid [1], ENCLUS [10], PROCLUS [9], and DOC [18] only find axis-parallel clusters. More recent algorithms such as ORCLUS [3] and 4C [7] can find clusters with arbitrarily oriented principle axes. However, none of them addresses our issue of finding nonlinear correlation clusters.

3. THE CURLER ALGORITHM
Our algorithm, CURLER, works in an interactive and top-down manner. It consists of the following main components.

1. EM Clustering: A modified expectation-maximization subroutine *EMCluster* is applied to convert the original dataset into a sufficiently large number of refined microclusters with varying orientations. Each microcluster M_i is represented by its mean value μ_i and covariance matrix Σ_i. At the same time, a similarity measure called co-sharing level between each pair of microclusters is computed.

---

[1] Note that as mentioned earlier, judging proximity by itself is a difficult task since the two microclusters could lie in different subspaces. We assume that the problem is solved here for ease of discussion.

[2] CURLER stands for CURve cLustERs detection.

2. Cluster Expansion: Based on the co-sharing level between the microclusters, a traversal through the microclusters is carried out by repeatedly choosing the nearest microcluster in the co-shared $\epsilon - neighborhood$ of a currently processed cluster. We denote this subroutine as *ExpandCluster*.

3. NNCO plot (Nearest Neighbor Co-sharing Level & Orientation plot): In this step, nearest neighbor co-sharing levels and orientations of the microclusters are visualized in cluster expansion order. This allows us to visually observe the nonlinear correlation cluster structure and the orientations of the microclusters from the NNCO plot.

4. According to the NNCO plot, users may specify clusters that they are interested in and further explore the local orientations of the clusters with regard to their global orientation.

In the next sections, we will explain the algorithm in detail and the reasoning behind it.

3.1 EM-Clustering

Like k-means, the EM-clustering algorithm is an iterative k-partitioning algorithm which improves the conformability of the data to the cluster model in each iteration and typically converges in a few iterations. It has various attractive characteristics that make it suitable for our purpose. This includes the **clustering model** it uses, the fact that it is a fuzzy clustering method, and its iterative refinement approach.

3.1.1 Clustering Model

In EM-clustering, we adopt a Gaussian mixture model where each microcluster M_i is represented by a probability distribution with density parameters, $\theta_i = \{\mu_i, \sum_i\}$, μ_i and $\sum_i$ being the mean vector and covariance matrix of the data objects in M_i respectively. Such a representation is sufficient for any arbitrary oriented clusters. Furthermore, the orientation of the represented cluster can be easily computed.

Banfield and Raftery [15] proposed a general framework for representing the covariance matrix in terms of its eigenvalue decomposition:

$$\Sigma_i = \lambda_i D_i A_i D_i^T, \qquad (1)$$

where D_i is the orthogonal matrix of eigenvectors, A_i is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_i, and λ_i is a scalar. D_i, A_i and λ_i together determine the geometric features (shape, volume, and orientation respectively) of component θ_i.

3.1.2 Fuzzy Clustering

Unlike ORCLUS and 4C in which each data object either belongs or not belongs to a microcluster, EM-clustering is a fuzzy clustering method in which each data object has a certain probability of belonging to each microcluster.

Given a microcluster with density parameters θ_k, we compute the probability of x's occurrence given θ_k as follows:

$$PR(x|\theta_i) = \frac{1}{\sqrt{(2\pi)^d |\sum_i|}} exp[-\frac{1}{2}(x - \mu_i)^T (\Sigma_i)^{-1}(x - \mu_i)], \qquad (2)$$

where x and mean vector μ_i are column vectors, $|\Sigma_i|$ is the determinant of Σ_i, and $(\Sigma_i)^{-1}$ is its inverse matrix.
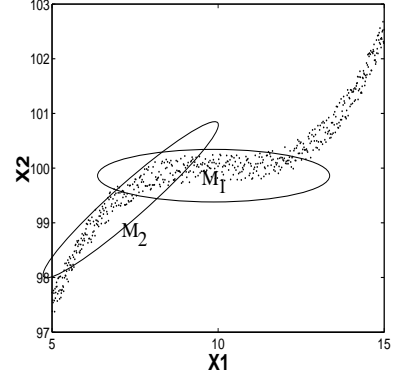

Figure 2: Co-sharing between Two Microclusters

Assuming the number of microclusters is set at k_0, the probability of x occurrence given the k_0 density distributions will be:

$$PR(x) = \sum_{i=1}^{k_0} W_i PR(x|\theta_i), \qquad (3)$$

The coefficient W_i (matrix weights) denotes the fraction of the database given microcluster M_i. The probability of a data object x belonging to a microcluster with density parameters θ_i can then be computed as:

$$PR(\theta_i|x) = \frac{W_i PR(x|\theta_i)}{PR(x)}. \qquad (4)$$

There are two reasons for adopting fuzzy clustering to form microclusters. First, fuzzy clustering allows an object to belong to multiple correlation clusters when the microclusters are eventually merged. This is entirely possible in real life datasets. For example, a hospital patient may suffer from two types of disease A and B, and thus his/her clinical data will be similar to other patients of disease A in one subset of features and also similar to patients of disease B in another subset of features. Second, fuzzy clustering allows us to indirectly judge the similarity of two microclusters by looking at the number of objects that are co-shared between them. More specifically, we define the following similarity measure:

DEFINITION 3.1. *Co-sharing Level*
The co-sharing level between clusters M_i and M_j is:

$$coshare(M_i, M_j) = \sum_{x \in D} [PR(M_i|x) * PR(M_j|x)], \qquad (5)$$

where x is a data object in the dataset D, $PR(M_j|x)$ and $PR(M_i|x)$ are the probability of object x belonging to microcluster M_i and microcluster M_j respectively. $PR(M_j|x)$ and $PR(M_i|x)$ are calculated according to Equations 4 and 2. □

Given each data object in the database, we compute the probability of the object belonging to both M_i and M_j at the same time and sum up these probabilities over all the data objects. In this way, the co-sharing level takes both the orientation and spatial distance of two microclusters into account without needing to explicitly determine their importance in computing the similarity. A high co-sharing value

between two microclusters indicates that they are very similar while a low co-sharing value indicates otherwise. As an example, consider Figure 2 where two microclusters, M_1 and M_2 are used to capture the bend in a cubic curve. Since M_1 and M_2 are neighboring microclusters, points that overlapped both of them will belong to both the Gaussian distributions and thus these points will increase the co-sharing level between them.

Note that this similarity measure is important here simply because we are handling nonlinear correlation clusters [3]. For linear correlation algorithms like ORCLUS and 4C, this measure is unnecessary as they can simply not merge two microclusters which are either too far apart or very dissimilar in orientation.

On the basis of our new *co-sharing level*, we will define the co-shared $\epsilon - neighborhood$ and nearest neighbor co-sharing level (NNC) for microclusters.

DEFINITION 3.2. *Co-shared $\epsilon - neighborhood$*
For a microcluster M_c, its co-shared $\epsilon - neighborhood$ refers to all the microclusters whose co-sharing level from M_c is no smaller than some non-negative real number ϵ: $\{\forall M_i |$ coshare(M_c, M_i) $\geq \epsilon\}$. $\square$

We will explain how these definitions will be useful in the section on cluster expansion later.

3.1.3 Iterative Refinement

Like the well-known k-means algorithm, EM-clustering is an iterative refinement algorithm which improves the quality of clustering iteratively towards a local optimal. In our case, the quality of clustering is measured by the log likelihood for the Gaussian mixture model as follows:

$$E(\theta_1, \ldots, \theta_{k_0} | D) = \sum_{x \in D} log[\sum_{i=1}^{k_0} W_i \cdot PR(M_i|x)] \quad (6)$$

The EM-clustering algorithm can be divided into two steps: E-Step and M-Step. In E-Step, the memberships of each data object in the microclusters are computed. The density parameters for the microclusters are then updated in M-Step. The algorithm iterates between these two steps until the change in the log likelihood is smaller than a certain threshold between one iteration and another. Such iterative change of memberships and parameters is necessary in order to break the catch-22 cycle described below:

1. Without knowing the relevant correlated dimensions, it is not possible to determine the actual neighborhood of the microclusters.

2. Without knowing the neighborhood of the microclusters, it is not possible to estimate their density parameters i.e., the mean vector and the covariance matrix of the microclusters.

---

[3]As an analogy, consider how soft metals like iron, copper, etc., can be easily bended because of their stretchable bond structures. Correspondingly, we can now 'stretch' data objects across microclusters because of fuzzy clustering so that the merged microclusters can conform to the shape of the nonlinear correlation clusters.

EMCluster(D, MCS, $\epsilon_{likelihood}$, $MaxLoopNum$)
1. Set the initial iteration Num. $j = 0$,
   initialize the mixture model parameters,
   W_i, μ_i^0 and Σ_i^0, for each microcluster $M_i \in MCS$.
2. (E-Step) For each data object $x \in D$:
   $$PR^j(x) = \sum_{M_i \in MCS} W_i PR^j(x|M_i),$$
   $$PR^j(M_i|x) = \frac{W_i * PR^j(x|M_i)}{PR^j(x)}, M_i \in MCS,$$
   $$W_i' = \sum_{x \in D} PR^j(M_i|x).$$
3. (M-Step) Update mixture model parameters for $\forall M_i \in MCS$:
   $$\mu_i^{j+1} = \frac{\sum_{x \in D}(x \cdot PR(M_i|x))}{\sum_{x \in D} PR(M_i|x)},$$
   $$\Sigma_i^{j+1} = \frac{\sum_{x \in D} PR(M_i|x)(x - \mu_i^{j+1})(x - \mu_i^{j+1})^T}{\sum_{x \in D} PR(M_i|x)}$$
   $$W_i = W_i'$$
4. If $|E^j - E^{j+1}| \leq \epsilon_{likelihood}$ or $j > MaxLoopNum$
   Decompose Σ_i for $\forall M_i \in MCS$ and return
   Else set $j = j + 1$ and go to 2.

   Note:
   E^j: the log likelihood of the mixture model at iteration j, $PR^j(x|M_i) = \frac{1}{\sqrt{(2\pi)^d | \Sigma_i^j |}} exp[-\frac{1}{2}(x - \mu_i^j)^T (\Sigma_i^j)^{-1}(x - \mu_i^j)]$.

Figure 3: EMCluster Subroutine

By sampling the mean vectors from the data objects and setting the covariance matrix to the identity matrix initially, the iterative nature of EM-clustering conforms the microclusters to their neighborhood through the iterations. Again, we note that our approach here is different from that of ORCLUS and 4C. ORCLUS does not recompute the microcluster center until two microclusters are merged, while 4C fixes its microclusters by gathering objects that are within a distance of ϵ of an object in full feature space. Our approach is necessary as we are finding more complex correlations. Incidentally, both ORCLUS and 4C should encounter the same catch-22 problem as us, but they are relatively unaffected by their approximation of the neighborhood.

The *EMCluster* subroutine is illustrated in Figure 3. First, the parameters of each microcluster M_i ($M_i \in MCS$) are initialized as follows: $W_i = 1/k_0$, $\Sigma_{M_i}^0$ is the identity matrix, and the microcluster centers are randomly sampled from the dataset. The membership probabilities of each data object x ($x \in D$), $PR(M_i|x)$, are computed for each microcluster M_i. Then the mixture model parameters are updated based on the calculated membership probabilities of the data objects. The membership probability computation and density parameters updating iterate until the log likelihood of the mixture model converges, or if the maximum number of iterations, $MaxLoopNum$, is reached. The output of the EM clustering is the means and covariance matrices of the microclusters, and also the membership probabilities of each data object in the microclusters. These results are passed on to the *ExpandCluster* subroutine.

```
ExpandCluster(MCS, ε, OutputFile)
1. Calculate the co-sharing level matrix;
2. M_c=MCS.NextUnprocessedMicroCluster
   C ={M_c} ;
3. N_C = neighbors(M_c, ε, MCS);
   M_c.processed = True;
   Output M_c to OutputFile;
   While |N_C| > 0 Do
     From N_C, remove nearest microcluster to C,
     and set it as M_c;
     M_c.processed = True;
     Output M_c and coshare(M_c,C) to OutputFile;
     Merge C and M_c to form new C_new;
     Update the co-sharing level matrix;
     C=C_new;
     N_C=N_C + neighbors(M_c, ε, MCS);
4. If there exist unprocessed microclusters goto 2;
End.
```

Figure 4: ExpandCluster Subroutine

3.2 Cluster Expansion

Having formed the microclusters, our next step is to merge the microclusters in a certain order so that the final nonlinear correlation clusters can be found and visualized.

DEFINITION 3.3. *Co-sharing Level Matrix*
The co-sharing level matrix is a $k_0 \times k_0$ matrix with its entry (i, j) representing the co-sharing level between microclusters M_i and M_j (coshare(M_i, M_j)). □

We calculate the co-sharing level matrix at the beginning of the cluster expansion procedure based on the membership probabilities $PR(M_i|x)$ for each data object x and each microcluster M_i. To avoid the complexity of computing $k_0 \times k_0$ entries for each data object x, we instead maintain for each x, a list of l_{top} microclusters that x is most likely to belong to. This reduces the number of entries update to l_{top}^2. We argue that x has 0 or near 0 probability of belonging to most of the microclusters and thus our approximation should be accurate.

As shown in Figure 4, the ExpandCluster subroutine first initializes the current cluster C as $\{M_c\}$, where M_c is the first unprocessed microcluster in the set of microclusters MCS. It then merges all other microclusters that are in the co-shared ϵ-neighborhood of M_c into N_C through the function call to neighbors(M_c, ϵ, MCS). M_c is then output together with its co-sharing level value with C. From among the unprocessed microclusters in N_C, the next M_c with the highest co-sharing level is found. C_{new} is then formed by merging M_c and C. We then update the co-sharing level matrix according to Equation 7.

$$coshare(C, M_k) = Max(coshare(C, M_k), coshare(M_c, M_k)), \quad (7)$$

where M_k is any of the remaining unprocessed microclusters.

C is then updated to become C_{new} and unprocessed microclusters in the co-shared ϵ-neighborhood of MC are added to N_C. This process continues until N_C is empty and then a C is re-initialized to another unprocessed microcluster by going to Step 2.

3.3 NNCO Plot

In the NNCO (Nearest Neighbor Co-sharing Level & Orientation) plot, we visualize the nearest neighbor co-sharing levels together with the orientations of the microclusters in cluster expansion order. The NNCO plot consists of a NNC plot above and an orientation plot below, both sharing the same horizontal axis.

3.3.1 NNC Plot

The NNC plot is inspired by the reachability plot of OPTICS [5]. The horizontal axis denotes the microcluster order in the cluster expansion, and the vertical axis above denotes the co-sharing level between the microcluster M_c and the cluster being processed C when M_c is added to C. We call this value the NNC (Nearest Neighbor Co-sharing) value of MC. Intuitively, the NNC plot represents a local hill climbing algorithm which moves towards the local region with the highest similarity at every step. As such, in the NNC plot, a cluster will be represented with a hill shape with the up-slope representing the movement towards the local high similarity region and the down-slope representing the movement away from the high similarity region after it has been visited. Note that an NNC level of 0 represents a complete separation between two clusters, i.e., the two clusters are formed from two sets of microclusters that do not co-share any data objects.

3.3.2 Orientation Plot

Below the NNC plot is the orientation plot, a bar consisting of vertical black-and-white lines. For each microcluster, there is a vertical line of d segments where d is the dimensionality of the data space, and each provides one dimension value of the microcluster's orientation vector, as defined below.

DEFINITION 3.4. *Cluster Orientation*
The cluster's orientation is a vector along which the cluster obtains maximum variation, that is, the eigenvector with the largest eigenvalue. □

Each dimension value y of the microcluster orientation vector is normalized to the range of [-127.5, 127.5] and mapped to a color ranging from black to white according to Equation 8.

$$Color(y) = [R(y + 127.5), G(y + 127.5), B(y + 127.5)] \quad (8)$$

Therefore, the darkest color ([$R(0)$, $G(0)$, $B(0)$], when $y = -127.5$) indicates the orientation parallel but opposite the corresponding dimension axis while the brightest color ([$R(255)$, $G(255)$, $B(255)$], when $y = +127.5$) indicates the orientation parallel and along the dimension axis. Gray ([$R(127.5)$, $G(127.5)$, $B(127.5)$], when $y = 0$) suggests no variation at all in the dimension. Obviously, similarly oriented microclusters tend to have similar patterns in the orientation plot. In this way, the clusters' specific subspaces can be observed graphically.

3.3.3 Examples

Figure 5 shows a quadratic cluster and a cubic cluster. The nonlinear cluster structures are detected successfully, as shown in the NNCO plots in Figure 6. According to Definition 3.1, the more similar in orientation the microclusters are, the larger the co-sharing level value they have. As our microclusters are assumed to be evenly distributed, the microclusters which are similar in orientations and close to each other have larger NNC values and tend to be grouped
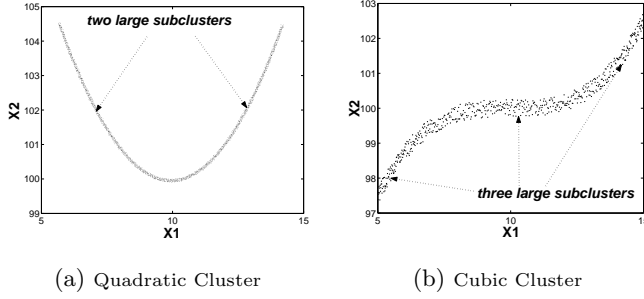
(a) Quadratic Cluster (b) Cubic Cluster

Figure 5: Quadratic and Cubic Clusters
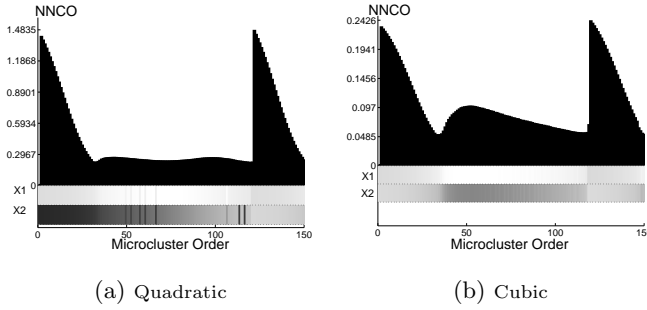

(a) Quadratic (b) Cubic

Figure 6: NNCO Plots

together. Here, the microcluster orientations are approximately the tangents along the curves. There are two humps, indicating two large subclusters of similar orientations in the quadratic NNC plot (Figure 6(a)). Likewise, there are three humps, indicating three large subclusters of similar orientations in the cubic NNC plot (Figure 6(b)).

Generally, the tangent projection along the quadratic curve in X_2 dimension increases from negative to positive while the tangent projection on the X_1 dimension increases and decreases symmetrically. The simple mathematic reasoning behind this is that, given the 2D quadratic curve

$$x_2 = a * (x_1 - b)^2 + c,$$

where $a > 0$, the changing ratio of the tangent slop, $x_2'' = 2 * a$, is a positive constant. The maximum tangent projection on the X_1 dimension is achieved when the tangent slope is 0. That is why we see in the orientation plot that as a whole, the bar color in dimension X_2 brightens continuously (tangent slope changes from negative to positive) while the bar color in dimension X_1 brightens first and darkens midway.

For the cubic curve

$$x_2 = a * (x_1 - b)^3 + c,$$

the tangent slope changes from positive to zero, then back to positive again. Again, as the tangent projection on dimension X_1 increases and decreases symmetrically while the tangent projection on dimension X_2 decreases and increases symmetrically. For this reason, the bar color in dimension X_1 brightens and darkens symmetrically while the bar color of dimension X_2 darkens and brightens symmetrically in the

orientation plot.

3.4 Top-down Clustering

Having identified interesting clusters from the orientation plot, it is possible to perform another round of clustering by focusing on each individual cluster. The reason for doing so is the observation that the orientation captured by the initial orientation plot could only represent the global orientation of the clusters.

As we know, each data object is assumed to have membership probabilities for several microclusters in CURLER. We define the **data members** represented by a discovered cluster C which consists of microcluster set MCS as the set of data objects whose highest membership probabilities are achieved in the microcluster among MCS, $\{\forall x | x \in D$ and $\exists M_c \in MCS$ such that $Max_{1 \leq i \leq k_0}\{PR(M_i|x)\} = PR(M_c|x)\}$. Based on the data members of cluster C, we can further compute the cluster existance space of C.

DEFINITION 3.5. *Transformed Clustering Space*
*Given the specified cluster C and l, we define the **transformed clustering space** as a space spanned by l vectors, denoted as ε_l^C, in which the sum of the variances along the l vectors is the least among all possible transformations. In other words, the l vectors of the transformed clustering space ε_l^C, are the l eigenvectors with the minimum eigenvalues, computed from the covariance matrix of the data members of C. We denote the l vectors as e_1, e_2, ..., and e_l, where l may be much smaller than the dimensionality of the original data space d.* □

Given the dimensionality of the original data space, d, a correlation cluster C_i, and l, we can further project data members of C_i, D^{C_i}, to the subspace $\varepsilon_l^{C_i}$ of l vectors ($\varepsilon_l^{C_i} = \{e_{i1}, e_{i2}, ..., e_{il}\}$) by transforming each data member $x \in D^{C_i}$ to $(x \cdot e_{i1}, x \cdot e_{i2}, ... x \cdot e_{il})$, where x and e_{ij} $(1 \leq j \leq l)$ are d-dimensional vectors. In this way, we obtain a new $l-D$ dataset and can carry on another level of clustering. Figure 7 shows the overview of our algorithm.

3.5 Time Complexity Analysis

In this section, we analyze the time complexity of CURLER. We focus our analysis on the EM-clustering algorithm and the cluster expansion since these two are the most expensive steps among the four.

● **EM Clustering:**
In the EM part, the algorithm runs iteratively to refine the microclusters. The bottleneck is Step 2, where the membership probabilities of each data object x for each microclusters $M_i \in MCS$ is calculated. The time complexity of matrix inversion, matrix determinant, and matrix decom-

CURLER(D, k_0, l_{top}, ϵ, $\epsilon_{likelihood}$, $MaxLoopNum$)
1. Randomly Sample k_0 number of seeds from D
   as MCS;
2. EMCluster(D, MCS, $\epsilon_{likelihood}$, $MaxLoopNum$);
3. Select one microcluster in MCS as c;
4. ExpandCluster(MCS, ϵ, $OutputFile$);
5. For any interesting cluster C_i
   Transform D^{C_i} into D_{new} in the subspace $\varepsilon_l^{C_i}$;
   CURLER(D_{new}, k_0', l_{top}, ϵ, $\epsilon_{likelihood}$, $MaxLoopNum$)
End.

Figure 7: CURLER

position is $O(d^3)$; thus, the time complexity of matrix operation for k_0 microclusters is $O(k_0 \cdot d^3)$. Besides, the time complexity of computing $PR^j(x|M_i)$ is $O(d^2)$ for each pair of x and M_i. For all data objects and all microclusters, the total time complexity of EM clustering is $O(k_0 \cdot n \cdot d^2 + k_0 \cdot d^3)$.

• **Cluster Expansion:**
The time complexity of computing the initial co-sharing level matrix is $O(n * l_{top}^2)$, as explained in Section 3.2. As there is no index available for CURLER due to our unique co-sharing level function, all the unprocessed microclusters have to be checked to determine the co-shared $\epsilon - neighborhood$ of the current cluster. So the time complexity of the nearest neighbor search for one cluster is $O(k_0)$ and the time complexity of the total nearest neighbor search is $O(k_0^2)$. Also, as the time complexity of each co-sharing level matrix update during cluster merging is $O(k_0)$, and there is maximum k_0 updates, the time complexity of the entire correlation distance matrix update is $O(k_0^2)$. As a result, the time complexity of the cluster expansion is $O(n \cdot l_{top}^2 + k_0^2)$.

4. EXPERIMENTAL ANALYSIS

We tested CURLER on a 1600 MHz PVI PC with 256M memory to ascertain its effectiveness and efficiency. We evaluated CURLER on a 9D synthetic dataset of three helix clusters with different cluster existence spaces, the iris plant dataset and the image segmentation dataset from the UCI Repository of Machine Learning Databases and Domain Theories [6], and the Iyer time series gene expression data with 10 well-known linear clusters [13].

4.1 Parameter Setting

As illustrated in Figure 7, CURLER generally requires five input parameters: $MaxLoopNum$, log likelihood threshold $\epsilon_{likelihood}$, microcluster number k_0, l_{top} and neighborhood co-sharing level threshold ϵ.

In all our experiments, we set $MaxLoopNum$ between 5 and 20, and $\epsilon_{likelihood}$ of 0.00001. The experiments show that it is quite reasonable to trade off a limited amount of accuracy for efficiency by choosing a smaller $MaxLoopNum$, a larger log likelihood threshold $\epsilon_{likelihood}$ and a smaller l_{top} ranging from 20 to 40.

The number of microclusters k_0 is a core parameter of CURLER. According to our experiments, there is no significant difference in performance when varying k_0. Of course, the larger the k_0, the more refined the NNCO plots we got. Unlike [3] where each data object is assigned to only one cluster, in CURLER, each data object is assumed to have membership probabilities for l_{top} microclusters. As a result, the performance of CURLER is not affected much by k_0.

The neighborhood co-sharing level threshold ϵ implicitly defines the quality of merged clusters. The larger ϵ indicates more strict requirement on microclusters' similarity in both orientation and spacial distance when expanding clusters; hence, the higher cluster quality we obtained. In our experiments, we set ϵ to 0. To get a rough clustering result for any positive ϵ, we simply moved the horizontal axis up along the vertical axis by a co-sharing level of ϵ in the NNCO plot. This is another advantage of our algorithm.

4.2 Efficiency

In this Section, we evaluate the efficiency of our algorithm

with a varying database size (n) and a varying number of microclusters (k_0) on the 9D (d=9) synthetic dataset. In our experiments, we fixed the maximum number of loop time $MaxLoopNum$ at 10, the log likelihood threshold $\epsilon_{likelihood}$ at 0.00001, the neighborhood co-sharing level threshold ϵ as 0, and the number of microcluster memberships for each data object l_{top} at 300. We varied either n or k_0. When n was varied, we fixed k_0 at 300. Likewise, we set n at 3000 when varying k_0. For the output results, we averaged the execution times of five runs under the same parameter setting. In general, CURLER performed approximately linearly with the database size and the number of microclusters, as illustrated in Figure 8. The high scalability of our algorithm shows much promise in clustering high-dimensional data.
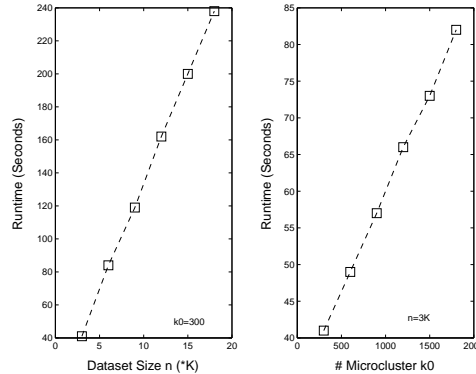

Figure 8: Runtime vs Dataset Size n and # Microclusters k_0 on the 9D Synthetic Dataset

4.3 Effectiveness

4.3.1 Synthetic Dataset

Because of the difficulty of getting a public high-dimensional dataset of well-known nonlinear cluster structures, we compared the effectiveness of CURLER with 4C on a 9D synthetic dataset of three helix clusters. The three helix clusters existed in dimensions $1 - 3$ (cluster 1), $4 - 6$ (cluster 2), and $7 - 9$ (cluster 3) respectively and the remaining six dimensions of each cluster were occupied with large random noise, approximately five times the data. Each cluster mapped a different color: red for cluster 1, blue for cluster 2, and yellow for cluster 3, as shown in Figure 9. Below is the basic generation function of helix, where $t \in [0, 6\pi]$,

$$
\begin{aligned}
x_1 &= c * t, \\
x_2 &= r * sin(t), \\
x_3 &= r * cos(t).
\end{aligned}
$$

The top-level NNC plot in Figure 10 shows that all the three clusters were identified by CURLER in the sequence of cluster 1, cluster 3 and cluster 2, separated by two NNC-zero-gaps. The top-level orientation plot further indicates the cluster existence subspace of each cluster, the gray dimensions. The noise dimensions are marked with irregular dazzling darkening and brightening patterns.

For a close look at the nonlinear correlation pattern of each cluster, we projected the data member into the corresponding cluster existence subspace of three vectors and performed sub-level clustering. Note that the vectors of the
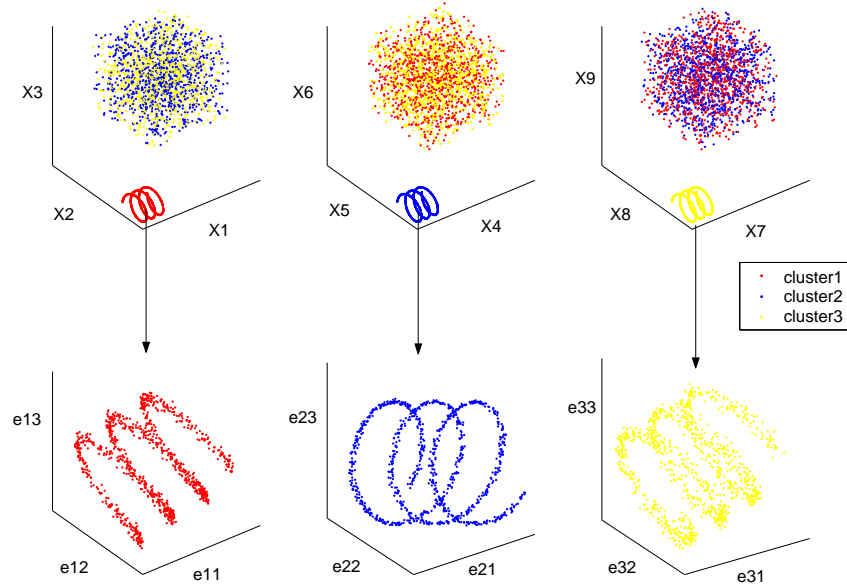
Figure 9: Projected Views of Synthetic Data in both Original Space and Transformed Clustering Spaces
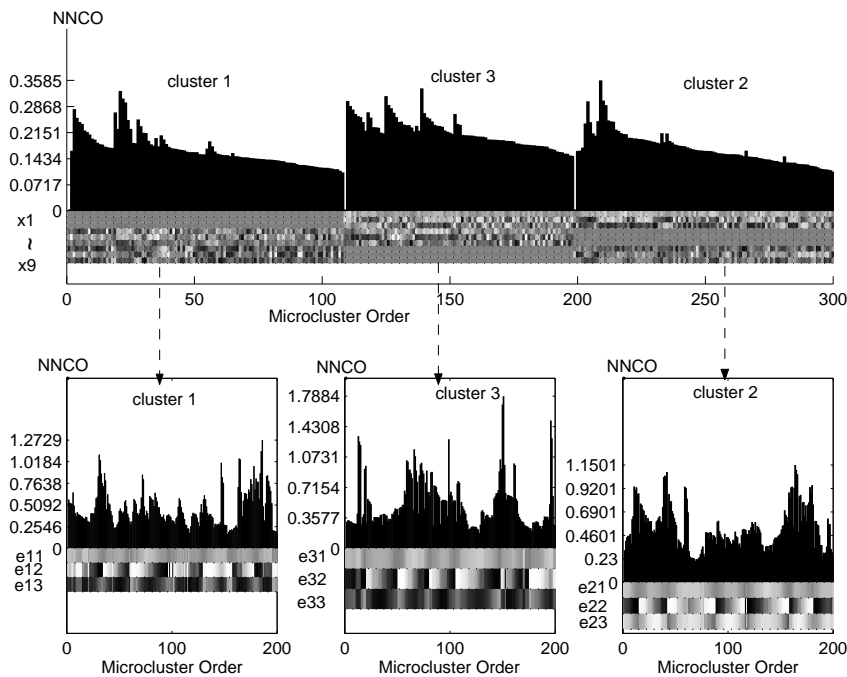

Figure 10: Top-level and Sub-level NNCO Plots of Synthetic Data

cluster existence subspace were NOT subsets of the original vectors. Since $|sin(t)|$ and $|cos(t)|$ had six cycles, when t varied from 0 to 6π, the sub-level NNCO plots show six cycles of shading and brightening orientation patterns in subspace dimensions e_{i1}, e_{i2}, and e_{i3} for each cluster i ($i = 1, 2,$ and 3).

As expected, 4C found no clusters although we set the correlation threshold parameter δ as high as 0.8. The changing orientation in the dataset does not exhibit the linear correla-

tion which 4C is looking for. In contrast, CURLER not only detected the three clusters but also captured their cycling correlation patterns and the subset of correlated features (Figure 10).

4.3.2 Real Case Studies

To have a rough idea of the potential of CURLER in practical applications, we applied the algorithm to three real-life datasets in various domains. Our experiments on the iris

plant dataset, the image segmentation dataset, and the Iyer time series gene expression dataset show that CURLER is effective for discovering both nonlinear and linear correlation clusters on all the datasets above. As the cluster structures of the first two public datasets have not been described, we will begin our discussion with the examination of their data distributions with the projected views. We will only report the top-level clustering results of CURLER here due to space constraint.

Based on our definition of the data members represented by cluster C in Section 3.4, we can infer the class cluster C mainly belongs to. We denote the inferred class label on the top of the cluster or subcluster in the NNCO plot.

4.3.2.1   Case 1: Iris Plant.

The iris plant dataset is one of the most popular datasets in pattern recognition domain. It contains 150 instances from three classes: Iris-virginica (class 1), Iris-versicolor (class 2) and Iris-setosa (class 3), 50 instances each. Each instance has four numeric attributes, denoted as X_1, X_2, X_3 and X_4. Figure 11 (a) shows the projected view of this data, where the blue points, green circle and red squares represent instances from class 1, 2 and 3 respectively. We can see that there are two large clusters: one consisting of instances of class 1 and the other consisting of instances from class 2 and class 3. The second cluster can further be divided into two subclusters, one composed of instances from class 2 and the other from class 3.

The microclusters constructed by the EMCluster subroutine are shown in Figure 12 (a). As can be seen clearly, the cluster expansion path traverses instances from class 1, class 2 and class 3 in an orderly manner. The NNCO plot of iris (Figure 13 (a)) visualizes two large clusters: one composed of 50 microclusters representing instances from class 1 and the second cluster composed of 100 microclusters representing instances from the other two classes. It is also noticeable that the second cluster is further divided into two subclusters (two humps) of 85 and 15 microclusters respectively. As illustrated in Figure 12 (a), the two subclusters mainly represent instances from class 2 and class 3 respectively. The different patterns of the clusters in the orientation plot suggest the corresponding different cluster existence subspaces. It is interesting that the microclusters in the same cluster or the same subcluster are very similar in orientation (very similar color patterns). Thus we can infer that the iris plant datasets have three approximately linear clusters, among which two with very similar orientations are close to each other.

4.3.2.2   Case 2: Image Segmentation.

The image segmentation dataset has 2310 instances from seven outdoor images: grass (class 1), path (class 2), window (class 3), cement (class 4), foliage (class 5), sky (class 6), and brickface (class 7). Each instance corresponds to a 3x3 region with 19 attributes. During dataset processing, we removed the three redundant attributes (attributes 5, 7, and 9 were reported to be repetitive with attributes 4, 6, and 8 respectively), and normalized the remaining 16 attributes to the range of [-5, 5]. The 16 attributes contained some statistical measures of the images, denoted as X_1, X_2, ..., X_{16}.

Figure 11 (b) shows the projected views on all dimensions. Figure 12 (b) is the projected view of our constructed microclusters on dimensions X_{14}, X_{15} and X_{16} in cluster expansion order.

Figure 13 (b) is the NNCO plot of the image dataset, which reveals the clustering structure accurately. Note that the image dataset is partitioned into three large clusters separated by NNC-zero-gaps. This is confirmed in our data projection views, Figure 11 (b.4) and (b.6), where we can see one large cluster composed of instances from class 1, one composed of instances from class 6, and another large cluster composed of mixed instances from the rest of the classes. The last cluster is nonlinear (Figures 11 (b.5) and (b.6)). The NNCO plot indicates that instances from the seven classes are well separated and fairly clustered.

The orientation plot further indicates that the clusters have their own subspaces; this is reflected in the different color patterns. However, some common subspaces also exist. For instance, we observe that the orientation plot on dimensions X_7, X_8, X_9, and X_{10} have synchronous color patterns, indicating synchronous linear correlations of the four attributes. As validated in Figures 11 (b.3) and (b.4), the three clusters approximately reside in the diagonal regions of dimensions X_7, X_8, X_9 and X_{10}. Another interesting phenomenon is that line X_1 is strongly highlighted (indicating large variation in X_1), line X_2 is partly highlighted (indicating positive orientation) and partly darkened (indicating negative orientation) while line X_3 is globally gray (indicating no variation at all in dimension X_3). With a closer look at Figure 11 (b.1), we see the answer: the three clusters distribute almost parallel with axis X_1 and have little variation in dimension X_3. The approximate gray of lines X_4, X_5, and X_6 also indicates little variation in the three dimensions. As a result of the nonlinear patterns in dimensions X_{11} to X_{16} (Figure 11 (b)), there are irregular color patterns in dimensions X_{11} to X_{16}.

Figure 14 depicts three interesting cluster structures discovered in the NNCO plot of the image dataset (Figure 13 (b)). First, the black-and-white cycling color pattern of microclusters 1-48 in dimensions X_{11}-X_{15} of the orientation plot is a vivid visualization of the nonlinear cluster structure of the corresponding instances of class 3 (Figure 14 (a)). Second, the synchronous three-vertical-bar pattern of microcluster 397-429 in both the NNC plot and the orientation plot, especially dimensions X_7-X_{10}, reveals three linear correlation clusters with diagonal orientations (Figure 14 (b)). The NNCO plot also indicates that the instances of class 7 can be partitioned into two big subclusters of consecutive microclusters, one represented by microclusters 49-82 and the other represented by microclusters 280-321 respectively. The plot also indicates that the later subcluster has a larger variation in dimensions X_{11}, X_{12}, and X_{13} (microclusters 280-321 have brighter colors in dimensions X_{11} and X_{12} of the orientation plot than microclusters 49-82). Again, this is verified in Figure 14 (c).

4.3.2.3   Case 3: Human Serum Data.

To verify the effectiveness of our algorithm, we also applied CURLER to a benchmark time series gene expression dataset, the Iyer dataset [13]. The Iyer dataset is a set of temporal gene expression data in response of human fibroblasts to serum, which consists of gene expression patterns of 517 genes across 18 time slots. [13] describes 10 linear correlation clusters of genes, denoted as 'A', 'B', ..., and 'J'. CURLER identified nine out of the reported ten clusters
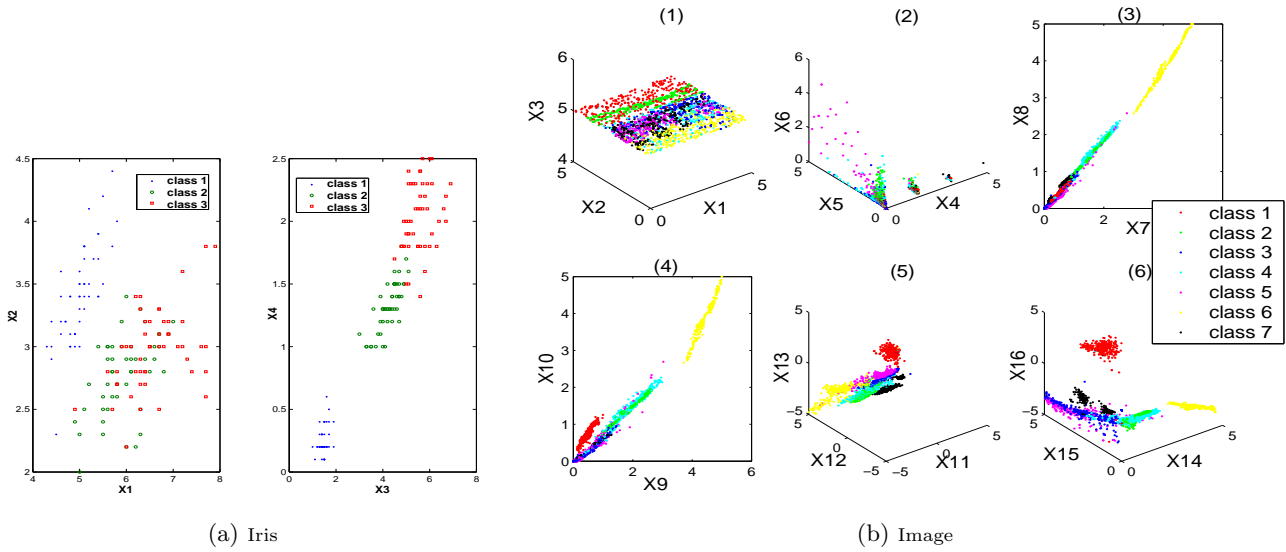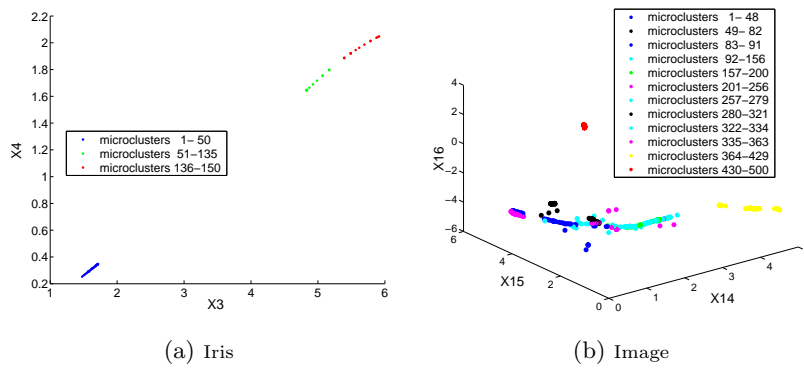
(a) Iris

(b) Image

Figure 11: Projected Views

(a) Iris

(b) Image

Figure 12: Constructed Microclusters

(a) Iris

(b) Image

Figure 13: NNCO Plots

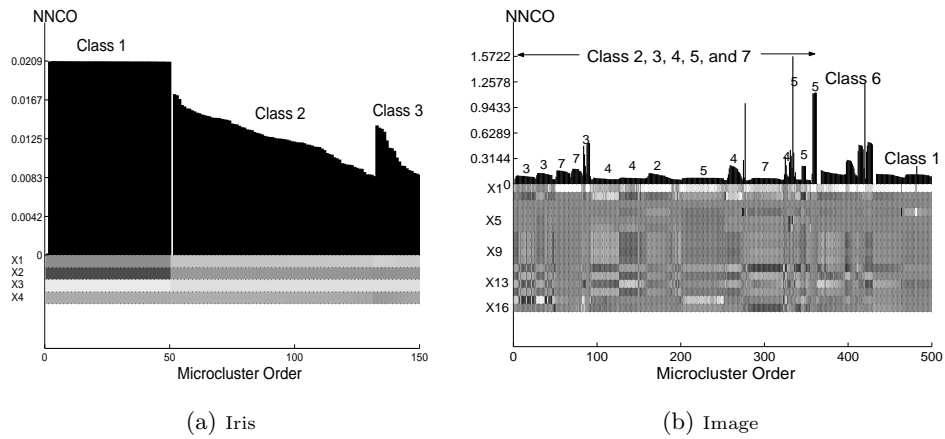(a) Microclusters 1-48   (b) Microclusters 397-429   (c) Microclusters 49-82 and 280-321
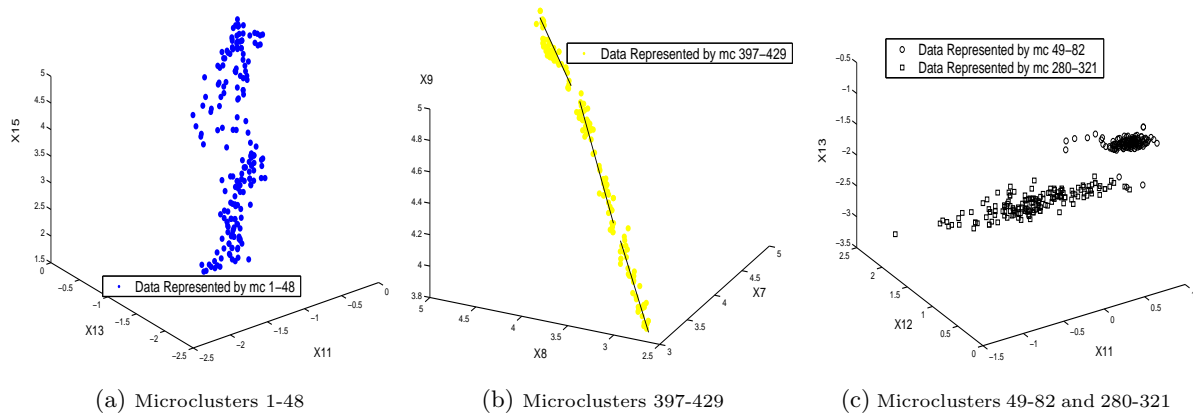
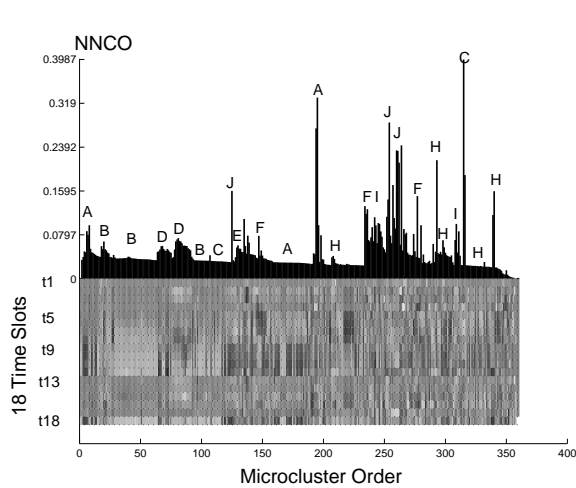**Figure 14: Cluster Structures Revealed by the NNCO Plots for the Image Dataset**
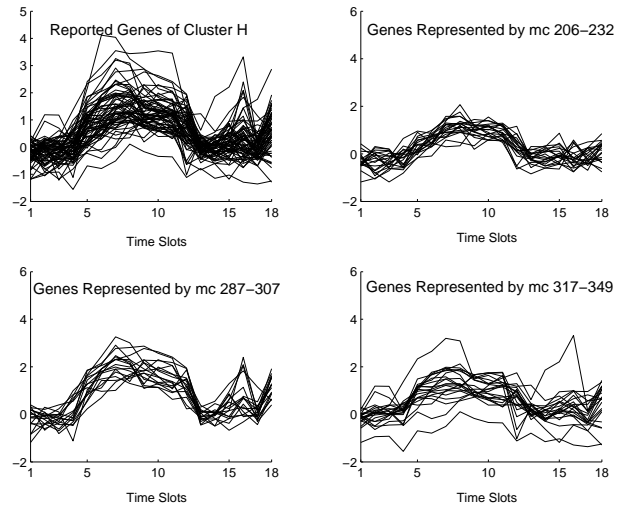

Figure 15: NNCO Plot of Iyer

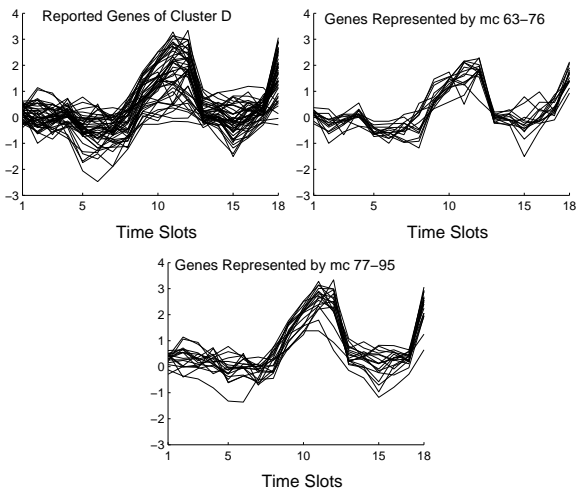**Figure 17: Discovered Subclusters for Cluster "H"**

Figure 16: Discovered Subclusters for Cluster "D"

successfully among the 517 genes (Figure 15); cluster 'G', consisting of 13 genes, was the exception. As can be seen, CURLER partitions the reported genes of cluster 'D' into two consecutive subclusters, represented by microclusters 63-76 and 77-95 respectively. Likewise, CURLER partitions the genes of cluster 'H' into three disjointed big subclusters of consecutive microclusters: 206-232, 287-307 and 317-349. The latter two big subclusters can be further partitioned at the sub-level as observed in the NNCO plot.

Figure 16 and 17 illustrate the temporal gene expression patterns across the 18 time slots of the genes in the above discovered subclusters. Apparently, the expression patterns of the genes in each subcluster are quite cohesive. Note that the expression patterns of genes in the two subclusters of cluster 'D' are different at time slots $t2$ and $t3$: those represented by microclusters 63-76 are negatively expressed while those represented by microclusters 77-95 are positively expressed. Besides, their variation at the two time slots are different, as detected by the NNCO plot. As for genes of the three subclusters of cluster 'H', their expression patterns are delicately different in time slots t9, t10, t11, and t12, as

shown in Figure 15 and verified in Figure 17.

5. CONCLUSIONS

In this paper, we have presented a novel clustering algorithm for identifying and visualizing nonlinear correlation clusters together with the specific subspaces of their existence in high-dimensional space. Almost no work has addressed the issue of nonlinear correlation clusters, let alone the visualization of these clusters. Our work is a first attempt, and it combines the advantage of density based algorithms represented by OPTICS [5] for arbitrary cluster shape and the advantage of subspace clustering algorithms represented by ORCLUS [3] for subspace detecting.

As shown in our experiments on a wide range of datasets, CURLER successfully captures the subspaces where the clusters exist and the nonlinear cluster structures, even when a large number of noise dimensions is introduced. Moreover, CURLER allows users to interactively select the cluster of their interest, have a close look at its data members in the space where the cluster exists, and perform sub-level clustering when necessary.

We plan to consider other variants to further improve the efficiency of CURLER, i.e., constructing some index structures to accelerate nearest neighbor queries based on the mixture model.

6. REFERENCES

[1] Hinneburg A. and Keim D. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of the 25th Int. Conf. on Very Large Data Bases*, pages 506 – 517, 1999.

[2] Hinneburg A. and Keim D.A. An efficient approach to cluster in large multimedia databases with noise. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, 1998.

[3] Yu P. S. Aggarwal C. C. Finding generalized projected clusters in high dimensional spaces. In *Proc. of ACM SIGMOD Conf. Proceedings*, volume 29, 2000.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of ACM-SIGMOD Int. Conf. on Management of Data*, pages 94–105, June 1998.

[5] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Int. Conf. on Management of Data*, pages 49–60, June 1999.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[7] Christian Bohm, Karin Kailing, Peer Kroger, and Arthur Zimek. Computing clusters of correlation connected objects. In *Proc. of ACM-SIGMOD Int. Conf. on Management of Data*, June 2004.

[8] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 9–15, Aug. 1998.

[9] Agrawal C. C., Procopiuc C., Wolf J. L., Yu P.S., and Park J. S. Fast algorithms for projected clustering. In *Proc. of ACM SIGMOD Int. conf. on Management of Data*, pages 61–72, 1999.

[10] C.H. Cheng, A.C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, 1996.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, Oregon, Aug. 1996.

[12] Patrik D Haeseleer, Xiling Wen, Stefanie Fuhrman, and Roland Somogyi. Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data. *Information Processing in Cells and Tissues*, pages 203–212, 1998.

[13] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J.Jr Hudson, M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.

[14] Han J. and Kamber M. *Data mining concepts and techniques*. Morgan Kaufmann, August 2001.

[15] Banfield J.D. and Raftery A.E. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, September, 1993.

[16] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.

[17] Kaufman L. and Rousseeuw P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

[18] C.M. Procopiuc, M. Jones, P.K. Agarwal, and M.T. M. A monte carlo algorithm for fast projective clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2002.

[19] J. Roy. A fast improvement to the em algorithm on its own terms. *JRSS(B)*, 51:127–138, 1989.

[20] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2323–2326, 2000.

[21] A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based clustering in large databases. In *Proc. 2001 Int. Conf. on Database Theory*, Jan. 2001.

[22] A. K. H. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. In *Proc. 2001 Int. Conf. on Data Engineering*, Heidelberg, Germany, April 2001.

[23] XU X., Ester M., Kriegel H-P., and Sander J. A distributed-based clustering algorithm for mining in large spatial databases. In *Proc. 1998 Int. Conf. on Data Engineering*, 1998.