Advanced Clustering Methods and Applications for Data Visualization

by

Bo Wu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
April 9, 2018

Keywords: clustering, grid, density, multi-dimensional scaling, relation, visualization

Approved by

Bogdan M. Wilamowski, Chair, Professor of Electrical and Computer Engineering
Thaddeus A. Roppel, Associate Professor of Electrical and Computer Engineering
Mark L. Adams, Assistant Professor of Electrical and Computer Engineering
Vitaly Vodyanoy, Professor of Anatomy, Physiology and Pharmacology
Erkan Nane, Associate Professor of Mathematics and Statistics

Abstract

Categorizing all sorts of data without any label information is one of the most important tasks in preprocessing raw data. To achieve that, an unsupervised learning technique, called clustering, is utilized. Generally, data are clustered based on their similarities or dissimilarities. However, in this big data era, the conventional clustering algorithms turn out to execute slow or perform with low clustering accuracy. Presented in this work include three clustering methods to accelerate clustering process and improve clustering accuracy.

The grid-based clustering is designed to fast process large amount of data. Combining with density-based clustering, the clustering method based on grid and density is capable of categorizing data with almost linear time complexity.

Inspired by natural mountain ridges, clustering method by finding data mountain ridges analyzes data layout as different mountain ridges. Each mountain ridge stands for one cluster. The fuzzy assignment is adopted to calculate data density, which is the data mountain height. It can find out the desired clusters without giving the number of clusters. More importantly, it has the capability of clustering data with complex shapes and noise.

Partitional clustering categorizes data into many microclusters. Known these microclusters, merging them into large clusters can provide us high accurate clustering results. Clustering method by analyzing density consistency and the minimum internal and external distance ratio takes the challenge to develop a strategy to determine whether to merge/

agglomerate microclusters or not so that clustering results with high clustering accuracy are obtained.

Besides, in the big data era, the demand for data visualization is increasing. As it is known that high dimensional data cannot be viewed by human beings, various dimension reduction approaches are explored to map/embed these data to a two-dimensional plane or three-dimensional space so that people can visualize them. Multi-Dimensional Scaling is a set of methods for dimension reduction purpose. Nevertheless, it suffers from mass data overlaps and no revelation of data relations. In this work, visualization algorithm combining with clustering is presented as well. It shows larger margins between clusters and the connection relations between data in the resulted figures.

In conclusion, the accomplishments in this work are as follows:

(1) Clustering time performance is boosted by applying grid technique;

(2) It is capable of clustering data with complex shapes and noise by finding data mountain ridges;

(3) High clustering accuracy is achieved by analyzing data density and the minimum internal and external distance ratio;

(4) A more accurate solution for MDS purpose is obtained by applying LM algorithm;

(5) Cluster separation regions are enlarged in the embedding results using density concentration;

(6) High dimensional data relations are revealed and illustrated in the embedding results.

Experiments were implemented on clustering and visualizing synthetic and real-world datasets to verify the effectiveness of the clustering and visualization methods introduced above.

Acknowledgments

I would like to express my sincere appreciation to my advisor, Dr. Bogdan M. Wilamowski, for his enduring tolerance of my dumbness, patient guidance, constant support and encouragement during my study at Auburn University. I feel very lucky to have such a highly respected advisor who taught a lot of knowledge in and out of his courses, gave me the freedom to think and explore, and helped me when I encountered difficulties. He taught me how to think creatively and work optimistically, which I think it is an invaluable treasure for my future career and life.

I would also like to thank Dr. Thaddeus A. Roppel, Dr. Mark L. Adams and Dr. Vitaly Vodyanoy for serving as my committee members and their support and suggestions for this work. Besides, I would like to thank Dr. Erkan Nane for the comments on my dissertation.

I would like to high praise God for all the blessings. With my heartfelt respect, I would like to thank my parents, Junxian Wu and Xiaolin Chen, for their continuous love, support and encouragement throughout my life. In addition, I would like to thank my Auburn Family, Yancy Carpenter, Joy Carpenter, Emmie, Jack, Julia, Mingjie Feng, Alfred and Hao Wu, for their love, concern, and encouragement during my study in Auburn. Last but not the least, I would like to thank my friends, including but not limited to, Vinny Zhu, Xuyu Wang, Jiao Jiao, Zerui Dong, Yiao Li, Yangyang Liu, Pinchen Cui, Jianliang Hao, Shen Zhang, Lingxiao Wang, Long Meng, Jordan Richardson and James Smith, for their love and inspirations.

Table of Contents

List of Tables

List of Figures

CHAPTER 1 INTRODUCTION

1.1 Background

With increasing demand for better understanding all sorts of raw data in this big data era, many unsupervised learning technologies are developed. Compared to supervised learning, where we have input data/variables and output data/variables and we use a supervised algorithm to learn the mapping function from the inputs to the outputs, unsupervised learning does not need any information about the output data/variables but still can process the input data/variables. Among those unsupervised learning approaches, two highlights are attracting our attention, which are clustering and embedding techniques.

As we know, one critical way to understand the interested information carried in the data is by finding the categories, or clusters of the data. Different clustering algorithms are proposed based on different definitions of clusters and different implementations. There are so many published clustering approaches, e.g. K-means [1], Hierarchical clustering [2], EM clustering [3], fuzzy clustering [4] etc. Although most of the clustering methods are generated years ago, clustering becomes even more powerful with the emergence of deep learning techniques. For example, in [5], authors utilize K-means clustering [1] as filters in convolutional neural networks (CNN) [6]. In [7,8], the proposed deep unsupervised clustering methods have achieved the state-of-the-art clustering performance. All those facts reach the conclusion that clustering is never too outdated to be used. In contrast, it is critical to handle clustering in big data era. Particularly, the demand for clustering algorithms with requirements of less computation time and ability to classify arbitrary-shape clusters rises.

Another important unsupervised learning technique is data embedding, specifically, multidimensional scaling/embedding. Multidimensional Scaling (MDS) [9] is one of the methods to reduce data dimensionality from high-dimensional space to low-dimensional space. MDS tries to map original high-dimensional data to low-dimensional projections with the expectation that data mutual distances are unchanged before and after mapping. After the concept of MDS was published, many MDS approaches are developed, e.g. Sammon mapping [10], Isomap [11], Local Linear Embedding [12] etc. One advantage of MDS methods is that the fundamental principle of MDS is simple so that we may come up with various solutions to it, which makes it as one of the main dimension reduction and visualization approaches.

1.2 Clustering Methods

Many clustering methods have been investigated in the literature for data preprocessing on grouping and categorizing. A brief description of conventional clustering methods is given below.

1.2.1 K-means Clustering

There are many clustering algorithms, but K-means clustering algorithm is the simplest and highly used one. In K-means clustering, cluster centers are the average locations of each category. So the basic idea of K-means clustering is keeping grouping data and updating the cluster centers until they never change. It is one of iterative clustering algorithms.

Suppose the data $\{x^1, x^2, \dots, x^n | x^i \in R^m, i \in [1, n]\}$, which have n data points and m dimension for each data point. Its desired cluster number is K, which means each data point i will be assigned to a cluster center c^i and c^i is one of those K centers. Now, the goal is to partition the data into K disjoint groups. The detailed K-means clustering algorithm is as follows.

Step 1: Place K center points $\{u^1, u^2, \ldots, u^K\}$ randomly among the data. These serve as initial cluster center points or centroids of the K clusters.

Step 2: Assign each data point to one of these K clusters based on its closeness (shortest distance from centroid) to the centroids of clusters. The distance used is usually Euclidean distance.

Step 3: When all data are assigned with a cluster, update the centroids of the clusters using equation (1).

$$u_k = \frac{\sum_{i=1}^{n} I\{c^i=k\} * x^i}{\sum_{i=1}^{n} I\{c^i=k\}} \quad (1)$$

Step 4: Repeat Step 2 and 3 until these K cluster centroids stop changing.

Because of its simplicity, K-means clustering is easily implemented. However, it suffers several drawbacks as follows.

A. It is hard to quantify the number of cluster centroids K;

B. Given the number of K, different initialization on these K centroids leads to different running time performance;

C. Although it is easy to understand, its time complexity is not linear but $O(tKnm)$, where *t* is the iteration number and *n* is the number of data points. How to deal with the situation when the dimensionality of data *m* is very large?

1.2.2 Hierarchical Clustering

In normal cases, the hierarchical clustering algorithm [2] we used is bottom-up algorithm. It treats each data point as a single cluster at the outset and then successively merges pairs of clusters until all clusters have been merged into a single cluster that contains all data points. This hierarchy of clusters is represented as a tree/dendrogram. The root of the tree is the unique

cluster that gathers all the samples, the leaves being the clusters with only one sample. There are three steps in hierarchical clustering algorithm.

Step 1. We begin by treating each data point as a single cluster. We then select a distance metric (e.g. Euclidean distance) that measures the distance between two clusters. As an example, we will use average linkage, which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.

Step 2. On each iteration, we agglomerate two clusters into one. The two clusters to be agglomerated are selected as those with the smallest average linkage. These two clusters have the smallest distance between each other and therefore are the most similar and should be agglomerated.

Step 3. Step 2 is repeated until we reach the root of the tree where we only have one cluster that contains all data points. In this way we can select how many clusters we want in the end, simply by choosing when to stop agglomerating the clusters.

Hierarchical clustering algorithm does not require specifying the cluster number and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric. A particularly good use of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of $O(n^3)$.

1.2.3 DBSCAN Clustering

As recognized as a very high-quality density-based clustering method, DBSCAN [13] is capable of determining clusters with arbitrary shapes and noise in data. With the idea that for

each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of data points, DBSCAN method makes several definitions before further processing.

1). Eps-neighborhood of a pattern: For a data point p, its Eps-neighborhood is:

$$N_{Eps}(p) = \{q \in D | Euc\_distance(p, q) \leq Eps\} \tag{2}$$

where D is the whole dataset. There should be at least a minimum number ($MinPts$) of data points in an Eps-region of a given pattern in a cluster.

2). Directly density-reachable: data point p is directly density-reachable from point q if

$$\begin{cases} p \in N_{Eps}(q) \\ \left|N_{Eps}(q)\right| \geq MinPts(core\ pattern\ condition) \end{cases} \tag{3}$$

Notice that directly density-reachable is not symmetric when cluster's core points and cluster's border points are involved.

3). Density-reachable: Data point p is density-reachable from point q if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

4). Density-connected: Point p is density-connected to point q if there is a data point o such that both p and q are density-reachable from o.

5). Cluster: A cluster C is a non-empty subset of the whole dataset satisfying the following two conditions:

$$\begin{cases} \forall p, q: if\ p \in C\ and\ q\ is\ density-reachable \\ \qquad from\ p,\ then\ q \in C. \\ \forall p, q \in C: p\ is\ density-connected\ to\ q. \end{cases} \tag{4}$$

6). Noise: For those data points that don't belong to any cluster, they will be regarded as noise.

To find a cluster, DBSCAN starts with an arbitrary data point p and retrieves all data points density-reachable from p. The two thresholds: Eps and $MinPts$ are set as global constants before processing. If p is a core data point, it yields a cluster. If p is a border pattern, then no data points are density-reachable from p and the next point will be processed. The overall average runtime complexity of DBSCAN is $O(n * \log(n))$, with the worst case of $O(n^2)$.

1.2.4 FSFDP Clustering

In FSFDP clustering method [14], it assumes that cluster centers are surrounded by neighbors with lower local densities and that they are at a relatively large distance from any data points with a higher local density. Based on that, there are four steps according to FSFDP clustering method.

Step 1: Calculate the pattern's local density ρ_i. The local density ρ_i of pattern i is calculated as

$$\rho_i = \sum_{j=1}^{np} \chi(d_{ij} - d_c) \tag{5}$$

where

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & otherwise \end{cases} \tag{6}$$

np is the total number of patterns in the dataset and d_c is a cutoff distance. Equation (5) will return the number of patterns with distances smaller than d_c to pattern i. The cutoff distance d_c is manually set at the beginning.

Step 2: Calculate the minimum distance δ_i between pattern i and any other pattern with a higher local density. The calculation of δ_i is:

$$\delta_i = \begin{cases} \min(d_{ij}), & \rho_j > \rho_i \\ \max(d_{ij}), & \rho_j < \rho_i \end{cases} \tag{7}$$

where $j = 1,2, \dots, np \; and \; j \neq i$. For the patterns that are local or global maxima in the density field, δ_i will be much larger than the typical neighbor distances. Thus, cluster center is classified as the pattern for which the value of δ_i is large.

Step 3: Generate the decision graph.

Aiming to choose the patterns with large local density ρ_i and large minimum distance δ_i as the cluster centers, the decision graph is generated with x-coordinate ρ_i and y-coordinate δ_i.

Step 4: Select the cluster centers.

It becomes visible to select the data with large local density ρ_i and large minimum distance δ_i as the cluster centers. After the selection of cluster centers in the decision graph, data points will be assigned into different clusters based on the minimum distances δs.

FSFDP is capable of clustering datasets with various shapes, e.g. *spiral* datasets. However, it's proven not feasible to classify datasets with more complicated shapes, e.g. chameleon datasets, in [15]. Besides, FSFDP requires the calculation of data mutual distances. The computational time complexity of FSFDP will be $O(n^2)$.

1.2.5 GMM-EM Clustering

Gaussian Mixture Model (GMM) [16] clustering algorithm is a model-based clustering approach, which assumes that the data points are generated from a mixture of Gaussians. For K-means clustering, it's not a model-based clustering but partitioning approach. But both of them can be solved using Expectation Maximization (EM) [17] technique.

- EM

EM is thought as a standard approach to find the maximum likelihood estimation. Maximum likelihood estimation [18] is to determine values for the parameters of a model. The parameter values are found such that they maximize the likelihood that the process described by

the model produced the data that were actually observed. EM is an iterative approach with two alternating steps: E step and M step, and eventually find the best parameter values. Suppose there is a dataset $\{x^{(i)}|x^{(i)}\epsilon R^m, i = n\}$ with n data points. Different from K-means clustering, we assume that their unknown/hidden labeling information $z^{(i)}$ of this dataset satisfies certain possibility distributions, and $z^{(i)}$ has *K* options. And,

$$p\left(z^{(i)} = j\right) = \emptyset_j > 0,\ \sum_{j=1}^{K} \emptyset_j = 1 \tag{8}$$

When $z^{(i)}$ is given as known, $x^{(i)}$ meets multivariable Gaussian distribution, which is $\left(x^{(i)}\middle|z^{(i)} = j\right)\sim N(\mu_j, \delta_j)$. The unknown parameter is $\theta = [\mu, \delta]^T$. Their joint probability distribution will be

$$p\left(x^{(i)}, z^{(i)}\right) = p(x^{(i)}|z^{(i)})p(z^{(i)}) \tag{9}$$

Now, the likelihood function will be

$$L(\theta) = L\left(x^{(1)}, x^{(2)}, \dots, x^{(n)}; \theta\right) = \prod_{i=1}^{n} p\left(x^{(i)}; \theta\right), \theta \in \Theta \tag{10}$$

It means that when the parameter set is equal to θ, the possibility of getting the data sample set of $\{x^{(i)}\}$ is equation (10). When θ varies, $L(\theta)$ varies and our goal is to maximize $L(\theta)$ as much as possible with respect to θ, which is our desired parameter estimation.

$$\hat{\theta} = argmax\ L(\theta) \tag{11}$$

As we can see, $L(\theta)$ is a combination of multiplications. So in order to make it easier to process, we can simply take log operation on $L(\theta)$ to turn it into the forms of additions, which is shown below.

$$\log\left(L(\theta)\right) = log \prod_{i=1}^{n} p\left(x^{(i)}; \theta\right) = \sum_{i}^{n} log\, p\left(x^{(i)}; \theta\right) \tag{12}$$

For EM algorithm, this parameter θ contains another unknown variable z. Our object becomes to find the proper θ and z so that $L(\theta)$ is the maximal.

$$L(\theta) = \sum_i log\, p\left(x^{(i)};\theta\right) = \sum_i log \sum_{z^{(i)}} p\left(x^{(i)}, z^{(i)};\theta\right) \quad (13)$$

$$= \sum_i log \sum_{z^{(i)}} Q_i\left(z^{(i)}\right) \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)} \quad (14)$$

$$\geq \sum_i \sum_{z^{(i)}} Q_i\left(z^{(i)}\right) \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)} \quad (15)$$

As we known that its second derivative is negative, equation (14) turns to be a concave function [19]. In equation (14), $\sum_{z^{(i)}} Q_i\left(z^{(i)}\right) \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)}$ is the expectation of $\frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)}$. According to Jensen's inequality [20], we can deduce equation (15). Now, in order to maximize $L(\theta)$, we will maximize its low bound of equation (15) and $L(\theta)$ will become larger accordingly. When its low bound reaches its maximal, and let $L(\theta)$ equals to the low bound, then we can have the solutions with respect to θ. According to Jensen's inequality, if $\frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)}$ is a constant variable (e.g. c), then $L(\theta)$ will equal to equation (15). For random variable z, its probability density function Q meets: $\sum_z Q_i\left(z^{(i)}\right) = 1$. So the sum of $p\left(x^{(i)}, z^{(i)};\theta\right)$ is equal to the constant, c. Now,

$$Q_i\left(z^{(i)}\right) = \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{\sum_z p\left(x^{(i)},z^{(i)};\theta\right)}$$

$$= \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{p\left(x^{(i)};\theta\right)}$$

$$= p\left(z^{(i)}|x^{(i)};\theta\right) \quad (16)$$

Overall, EM algorithm has the procedure of:

Step-E: Use the posteriori probability from the initial or last iteration as the parameter estimation of hidden variable.

$$Q_i\left(z^{(i)}\right) = p\left(z^{(i)}|x^{(i)};\theta\right)$$

Step-M: Maximize likelihood function $L(\theta)$ to gain an updated parameter:

$$\theta = \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i\left(z^{(i)}\right) \frac{p\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)} \tag{17}$$

Keep alternating computation between Step-E and Step-M, then we will get the parameter θ, which maximizes the likelihood function.

- GMM

Each GMM is consist of K Gaussian distributions and each Gaussian is call one component. For one component, it has an "ellipse" shape on 2D plane and its probability density function is

$$N(x,\mu,\Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^T(x-\mu)\Sigma^{-1}\right] \tag{18}$$

where μ is data expectation and Σ is the variance. But in many cases, data normally don't satisfy single Gaussian distribution and they have complex shapes. Then, there comes GMM clustering method. GMM is the linear combination of K components, where K is the number of clusters. The probability density function of GMM is

$$\begin{aligned} p(x) &= \sum_{k=1}^{K} p(k)p(x|k) \\ &= \sum_{k=1}^{K} \pi_k \mathrm{N}(\mathrm{x}|\mu_k,\Sigma_k) \end{aligned} \tag{19}$$

The log likelihood function of GMM will be as follows.

$$\log(L) = \sum_i^N log \sum_{k=1}^{K} \pi_k \mathrm{N}(\mathrm{x}|\mu_k,\Sigma_k) \tag{20}$$

In order to maximize the probability density function of GMM, we just need to maximize its likelihood function. Now, the problem becomes to be exactly maximum likelihood estimation, which can be solved using EM method. The procedure of GMM algorithm is shown below.

Step-E: Estimate the probability that data *i* belongs to the *k*-th component.

$$\gamma(i,k) = \frac{\pi_k \mathrm{N}(x_i|\mu_k,\Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathrm{N}(x_i|\mu_k,\Sigma_k)} \tag{21}$$

where $\mathrm{N}(x_i|\mu_k,\Sigma_k)$ is the posterior probability:

$$\mathrm{N}(x_i|\mu_k,\Sigma_k) = \frac{1}{(2\pi)^{n/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T(x-\mu)\Sigma^{-1}\} \tag{22}$$

Step-M: Update parameters.

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma(i,k) x_i \tag{23}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma(i,k)(x_i-\mu_k)(x_i-\mu_k)^T \tag{24}$$

$$\pi_k = \frac{N_k}{n} \tag{25}$$

where

$$N_k = \sum_{i=1}^{n} \gamma(i,k) \tag{26}$$

Keep iteratively computing between Step-E and Step-M until likelihood function converges and these components are the K cluster centers accordingly. For each iteration, Step-E has the time complexity of $O(nmK)$, where m is the dimensionality of data x. The time complexity of Step-M is $O(nm)$.

1.2.6 Spectral Clustering

Spectral clustering method [21] utilizes graph connection to separate/cut different clusters. The connections between data points are weighted based on their similarities and the goal is to

minimize the connection weights between clusters. In spectral clustering, similarity/weights matrix is decomposed to perform dimensionality reduction before clustering. Specifically, it takes use of spectrum/eigenvalues of the similarity/weights matrix to reduce dimension to fewer dimensions. This symmetric similarity matrix E can be derived to Laplacian matrix L and it's encouraged to adopt Laplacian transformations. The element e_{ij} in similarity matrix E denotes the weight between graph point i and j. For the diagonal, it's all zeros in E. Then Laplacian matrix L will be

$$L = D - E \tag{27}$$

where D is the diagonal matrix and $D_{ii} = \sum_j E_{ij}$.

Another commonly applied strategy to calculate Laplacian matrix is using normalization [21], which

$$L^{norm} = I - D^{-1/2} E D^{1/2} \tag{28}$$

where matrix I is the identity matrix with ones on the main diagonal and zeros elsewhere. The procedure of spectral clustering algorithm is:

Step 1：use the data to generate the similarity matrix based on graph analysis;

Step 2：get the normalized Laplacian matrix;

Step 3：generate K smallest eigenvalues and the corresponding eigenvectors, where K is the desired cluster number;

Step 4：apply K-means clustering to cluster these K eigenvectors (smaller dimensions) and return the final clustering results.

Basically, spectral clustering uses matrix operation to reduce dimensionality nonlinearly and the adopt K-means clustering approach. In Step 1, the time complexity will be $O(n^2)$ and we have to store the matrix as well.

1.2.7 Fuzzy Clustering

Fuzzy clustering, or soft clustering, is a form of clustering in which each data point belongs to more than one cluster. As comparison, in non-fuzzy clustering methods, each data point can only belong to exactly one cluster. In fuzzy clustering methods, we need to define the membership function, which is used to assign grades to each data point. These membership grades indicate the degree to which data points belong to each cluster. Many clustering methods use fuzzy clustering as part of their algorithms, for example fuzzy c-means clustering method [22], GMM clustering method etc.

1.2.8 Clique Clustering

The algorithm of Clustering In QUEst [23] is developed to solve high-dimensional data clustering problems. It is a grid-based clustering method, utilizing the property of dense grid units connection in data subspace. It discretizes the data space through a grid and estimates the density of grid units by counting the number of data points in a grid cell. A grid unit is dense if the fraction of total data points contained in the grid unit exceeds the threshold. A cluster in Clique is defined as a maximum set of connected dense grid units. Two input parameters of Clique clustering are the size of grid and a global density threshold for clusters. The major difference between this algorithm and other density-based and/or grid-based clustering approaches is that this method also detects subspaces of the highest dimensionality as high-density clusters exist in those subspaces. The major steps of Clique clustering are shown as follows.

Step 1: Partition each subspace that has dimension 1 into the same number of equal length intervals.

Step 2: Find dense regions (clusters) in each subspace and generate their minimal descriptions.

Step 3: Use dense regions to find the promising candidates on 2D plane based on the Apriori principle [24].

Step 4: Repeat Step 1-3 in level-wise manner in higher dimensional subspaces.

In Clique clustering algorithm, subspaces and clusters in the subspaces can be identified. It runs fast with linear scaling to the size of data. The clustering quality of Clique depends on the choice of grid size. Let k be the highest dimensionality of any dense unit, then its running time is $O(c^k + n * k)$ with a constant c.

1.3 Data Visualization Methods

For real life data, they are usually of many attributes, much more than three. As human beings, we can not see the data by our eyes in the space over three. So it is necessary to develop an algorithm to map high-dimensional data into a 2D/3D space so that we can visualize them and get a better understanding of them. Especially in this big data era, it's critical to build fast and efficient models to visualize big data.

For this task of visualizing high-dimensional data in low-dimensional space, it demands to reduce dimensionality. A fast solution to dimensionality reduction is by matrix operations, for example, Principle Component Analysis (PCA) [25], Non-negative Matrix Factorization (NMF) [26], Linear Discriminant Analysis (LDA) [27] etc. They take use of all kinds of matrix operations and find the most important or related several elements in the analyzed matrix so that reduce the number of needed elements. However, simply by finding the most important

dimensions/attributes of data and neglecting rest dimensions/attributes will definitely lose certain information. Nowadays, more and more publications show the efficiency to use optimization methods to accomplish this task. For example, Stochastic Neighbor Embedding (SNE) [28] and its improved extension t-distribution SNE (t-SNE) [29] present a solution for data visualization by preserving the probability distributions of high-dimensional and low-dimensional data.

Another optimization based visualization technique is Multi-Dimensional Scaling (MDS) [9,30]. MDS is a family of different methods in order to visualize high-dimensional data in a low-dimensional space, e.g. 2D plane etc. An MDS algorithm aims to place each object in high-dimensional space such that the between-object distances are preserved as well as possible.

1.3.1 MDS

MDS arranges the low-dimensional data points so as to minimize the discrepancy between the pairwise distances in the original high-dimensional (HD) space and the pairwise distances in the low-dimensional space, e.g. 2D. The general form of cost function of MDS algorithms is show as follows, which is solved using optimization method.

$$Cost = \sum_{i<j}(d_{ij} - D_{ij})^2 \tag{29}$$

where d_{ij}, D_{ij} are the pairwise distance between data point i and j in 2D space and HD space, respectively. In general, Euclidean distance is used to measure the dissimilarity/distance between data. The Euclidean distance between data x_i and x_j is:

$$d_{ij} = \sqrt{\sum_{k=1}^{m}(x_i^{(k)} - x_j^{(k)})^2} \tag{30}$$

where m is the number of data dimensions/attributes.

- Metric MDS

Metric MDS is a superset of MDS that generalizes the optimization procedure to a variety of loss functions and distances with weights and so on. Metric MDS minimizes the cost function called "*Stress*" which is a residual sum of squares:

$$Stress = \sqrt{\sum_{i<j}(d_{ij} - D_{ij})^2} \tag{31}$$

It is basically the same as equation (29).

- Non-metric MDS

In contrast to metric MDS, non-metric MDS finds both a non-parametric monotonic relationship between the dissimilarities/distances and the Euclidean distances. The key factor that non-metric MDS is different from metric MDS is the preservation of distance relationships. Typically, the relationship can be found using isotonic regression: let x denote the vector of proximities, $f(x)$ a monotonic transformation of x, which also contains the relationship of distances D; then coordinates have to be found, that minimize the so-called *stress*,

$$Stress = \sqrt{\frac{\sum(f(x)-D)^2}{\sum D^2}} \tag{32}$$

A few variants of this cost function exist. MDS algorithms automatically minimize stress in order to obtain the MDS solution.

In the following, several commonly used MDS techniques are described.

1.3.2 Sammon Mapping

Sammon mapping [10] is MDS approach expecting to maintain all the distances from one data point to the others, which turns to be a global optimization problem. Sammon mapping is a generalization of the usual metric MDS. Its stress function, which is to be minimized, is shown as follows.

$$Stress = \frac{1}{\sum_{l<k} D_{lk}} \sum_{i<j} \frac{(d_{ij} - D_{ij})^2}{D_{ij}} \quad (33)$$

In the stress function, it normalizes the squared-errors in pairwise distances by using the distance in the high-dimensional space. As a result, Sammon mapping preserves the small D_{ij}, giving them a greater degree of importance in the fitting procedure than for larger values of D_{ij}. In other words, it puts too much emphasis on getting very small distances exactly right. As a consequence, it's slow to optimize and also gets stuck in different local optima each time.

1.3.3 Isomap

Isomap [11] is also a global MDS and metric MDS technique, desiring to keep every single data-wise distance the same as in the high-dimensional space and low-dimensional space. The difference between other MDS and Isomap is that common MDS performs low-dimensional embedding based on the pairwise distance between data points, which is generally measured using straight-line Euclidean distance. Isomap is distinguished by its use of the geodesic distance induced by a neighborhood graph embedded in the traditional MDS. The geodesic distance between two data points is computed as the sum of small neighbor-point distances along the shortest path between these two data points. Technically, geodesic distances can be calculated using Dijkstra's algorithm [31] or Floyd-Warshall algorithm [32]. Once the geodesic distances are obtained, Isomap proceeds embedding using simple matrix operations or traditional MDS scheme. Isomap incorporates manifold structure in the resulting embedding.

1.3.4 Local Linear Embedding

Different from Sammom mapping and Isomap, Local Linear Embedding (LLE) [12] is a neighborhood based MDS approach, which emphasizes to preserve data neighbor/local information.

In LLE, the idea is to make the local configurations of data points in the low-dimensional space resemble the local configurations in the high-dimensional space. In the first, data point *i*'s near neighbor points are found, which is denoted as $N(i)$. Next, the reconstruction weights w are calculated by minimizing the stress function in equation (34), in which x denotes the data point in the high-dimensional space. Note that the sum of w_{ij} is limited to 1. If data point *j* is not a neighbor of data point *i*, w_{ij} is set to be 0.

$$Stress = \sum_i \left\| x_i - \sum_{j \epsilon N(i)} w_{ij} x_j \right\|^2, \ \ \sum_{j \epsilon N(i)} w_{ij} = 1 \tag{34}$$

In the last step, the same weights w_{ij} that reconstructs the *i*-th data point in the high dimensional space will be used to reconstruct the same point in the low dimensional space. A neighborhood preserving map is created based on this idea. Each data point x_i in the high dimensional space is mapped onto a data point y_i in the low dimensional space by minimizing the cost function of (35).

$$Stress = \sum_i \left\| y_i - \sum_{j \epsilon N(i)} w_{ij} y_j \right\|^2 \tag{35}$$

where $N(i)$ is the set of data point *i*'s neighbors.

1.3.5 LAMP

In contrast to other MDS, LAMP [33] is one of the MDS techniques based on landmarks. It has two main procedures as follows.

Step 1 - Control data points mapping or landmark data points mapping.

Step 2 - Affine mapping.

In the first step of LAMP, certain number of control points are selected. Based on these landmark points, the rest data points will be projected to the low-dimensional space. Given data point x,

$x \in \mathbb{R}^m$, LAMP method maps x to the low-dimensional space by finding the best affine transformation $f_x(p) = pM + t$ that minimizes

$$\sum_i \alpha_i \|f_x(x_i) - y_i\|^2, \ subject\ to\ M^T M = I \tag{36}$$

where matrix *M* and vector *t* are the unknowns, x_i is the *i*-th element of the subset of control points, *I* is the identity matrix and α_i are scalar weights defined as:

$$\alpha_i = \frac{1}{\|x_i - x\|^2} \tag{37}$$

The minimization problem (36) is a typical example of the Orthogonal Procrustes Problem [34], whose solution can be obtained using matrix singular value decomposition (SVD) [35].

LAMP can fast map high-dimensional data. However, different landmarks give significantly different MDS results.

1.4 Research Objectives

The objectives for developing clustering methods are as follows:

- Guarantee the speed of fast processing data.
- Ensure the efficiency of dealing with data that have complex shapes.
- The ability to detect and remove noise from data.
- Improve clustering accuracy.

And, the objectives for visualization algorithm design are as follows:

- Find the solution for MDS problem more accurately than common local solutions.
- Make the cluster separation regions more clearly.
- Reveal the data relations in the embedding results.

1.5 Organization of the Dissertation

The focus of this dissertation is the development of an advanced fast clustering approach and a novel MDS technique for data and relation visualization based on advanced clustering method. Each chapter in this dissertation is organized as below.

Chapter 1 introduces the background of this research and presents the research objectives in this work. It also provides the clustering methods and MDS techniques used for references.

Chapter 2 describes the clustering algorithm based on grid and density peaks. By introducing grid approach and fuzzy/soft cluster assignment, the data densities are well presented using the described clustering algorithm. Users can interactive with this clustering algorithm by manually choosing the cluster centers in the decision graph so that the clustering results can be obtained with the selected data points as the desired cluster centers.

Chapter 3 explores a fast density and grid based clustering method for data that has complex shapes and noise. By following the directions of data density ridges, the main body of different clusters can be found automatically and efficiently. Cluster border points and noise are explicitly explained and treated accordingly so that no data points are left un-clustered and no noise are clustered. Experimental results show that by applying this clustering strategy, it is possible to cluster data with arbitrary shapes and noise.

Chapter 4 presents a clustering method by analyzing density consistency and minimum internal and external distance ratio. The new definition of cluster internal and external distance ratio is proposed. Partitioned clusters are generated by applying density-based clustering approach. Besides, density consistency between partitioned clusters is analyzed. Accuracy of clustering data with complex shape can be improved by adopting our method.

Chapter 5 investigates data visualization technique with second-order optimization. Second-order optimization method is capable of offering better convergence results compared with first-order approaches. The multi-dimensional scaling (MDS) results are augmented with the second-order optimization trying to reduce the distance errors in MDS. By involving density based clustering concept, data moves towards the supposed inner-cluster data and, therefore, better separations between clusters are formed, which indicates a better performance of the proposed method over other MDS approaches.

Chapter 6 introduces how to apply data visualization method to reveal data relations. According to density based clustering approaches, data are related to each other based on their densities and distances. It turns out that by adding this kind of relations to the visualization results, users can receive even more useful information/knowledge from the embedded outcomes.

Chapter 7 presents conclusions and suggestions for future work.

CHAPTER 2 CLUSTERING ALGORITHM BASED ON GRID AND FINDING DENSITY PEAKS

Clustering or categorizing an unprocessed data set is essential and critical in many areas. Much success has been published, which first needs to calculate the mutual distances between data points. It suffers from considerable computational costs, preventing the state-of-the-art methods such as the clustering method by fast search and find of density peaks [14] from applying into real life (e.g., with thousands of data points). In this chapter, an efficient grid-based clustering (GBC) method by finding density peaks is described. It keeps the advantage of the friendly interactive interface in the FSFDP, at the meantime, decreases enormously the computation complexity. The time complexity of the FSFDP is $O(n(n-1)/2)$ while our method decreases it to $O(n * grid\_size)$, where np is the number of data points and the size of grid is always much smaller than n so that the time complexity of our approach is almost linearly proportional to n. The presented GBC method by finding density peaks was able to calculate the densities and categorize datasets within much less time, which makes the density-peak-based algorithm practical. By using the presented algorithm, it was possible to cluster high-dimensional data sets as well. The GBC method by finding density peaks was successfully verified in clustering several datasets, which are commonly used to test clustering algorithms in published articles. It turned out that the presented method is much faster and efficient in clustering datasets into different categories than the conventional density-based ones, which makes the proposed method more preferable.

2.1 Grid Based Clustering

Although the idea of grid-based clustering was proposed almost twenty years ago as described tens years ago, for example, BANG-clustering method [23, 36], etc., it is even more useful and powerful today. That's because the grid was used either to pre-process the patterns in the first stage or simply to represent some time-costly approaches, i.e. K-means [1], hierarchical clustering [2], etc., in the past. In the traditional approaches, the grid is divided into rectangular but not limited to rectangular segments with different sizes, which conceals the overwhelming capability of the universal grid with universal size.

Here, we define the term of the standard grid as the grid with segments in it with the length of 1. The standard grid in a two-dimensional (2D) space is the grid with squares of length 1 in it. The standard grid in a three-dimensional space is the grid with the cube of length 1 in it. The segments and intersection points in the standard grid are called cells and nodes, respectively. The number of cells in one dimension is called the size of the grid in that dimension. In this chapter, the standard grid with different sizes will be tested.

2.2 Density Peaks Based Clustering

This chapter presents another attempt to cluster patterns with the basic idea of density peaks in FSFDP [14]. It is much faster to categorize patterns even with nonspherical shapes than FSFDP without losing the ability to cluster high-dimensional patterns, which makes it practical and efficient to be used in real life.

2.2.1 Fuzzy Clustering

In all sorts of fuzzy clustering methods (e.g. [4, 22]), each data can belong to more than one category. Based on predefined fuzzy functions, each data will have different possibilities to

be treated as different clusters at the same time. In most cases, different definitions of fuzzy functions generate different fuzzy clustering results.

2.2.2 Density Based Clustering

The density based clustering approach that is discussed in this part is originally from FSFDP [14]. Compared to other density based clustering methods, e.g. [13], FSFDP method needs much less parameter settings and, at the same, provides users an interactive graph that enables users to select cluster centers from it. The detailed algorithm procedure is well described in Section 1.2.4.

2.3 Clustering Algorithm Based on Grid and Density Peaks

A grid-based clustering method by finding the density peaks will be discussed in this part. It is efficient and much faster as the FSFDP is concerned. The FSFDP can not be applied to the presented method directly, but it can be extended to it. There are three steps to conduct the presented algorithm: (I) normalizing and expanding the original data set into the grid; (II) computing the node's local density; (III) generating the decision graph.

2.3.1 Normalization and Expansion

At first, original data set is normalized in each dimension and then scaled into $[1, N\_grid]$ range grid, where N_grid is the size of grid in each dimension. This simplifies the calculations of nodes local densities.

2.3.2 Calculation of Node's Local Density

Instead of computing pattern's local density, the node's local density will be used. In this proposed method, nodes are only located in the grid with integer coordinates. Two options can be adopted to determine the node's local density: 1) the hard decision using round approximation

and 2) the soft decision using fuzzy type approximation. For a better view of the node's local density, spline technique [37] will be applied between nodes.

A. Hard Decision using Round Approximation

The round function is:

$$round(x) = \begin{cases} floor(x), & x - floor(x) < 0.5 \\ floor(x) + 1, & x - floor(x) \geq 0.5 \end{cases} \tag{38}$$

where $floor(x) = \max(m), m \in \mathbb{Z}\ and\ m \leq x$, and $\mathbb{Z}$ is the set of integers (negative, positive and zero). If the fractional portion $(x - floor(x))$ of x is 0.5 or greater, the argument is rounded to the next higher integer. If the fractional portion $(x - floor(x))$ of x is less than 0.5, the argument is rounded to the next lower integer. When using round approximation to compute the node's local density, each individual pattern contributes only to its nearest node's density by 1. The nearest node of one individual pattern is found by applying round function in each dimension. After conducting round approximation into all patterns, the local densities for all nodes are available.

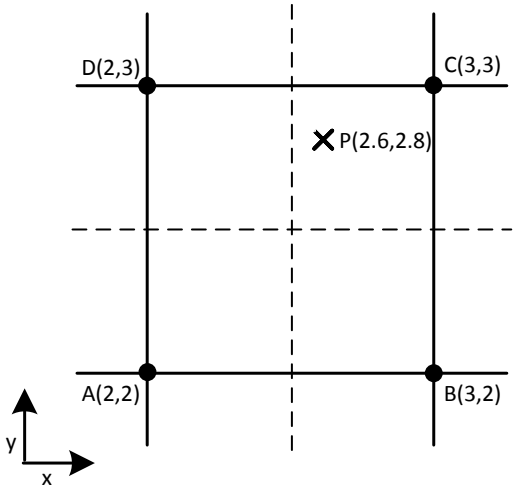

Figure 2.1 An example of node's local density calculation on a 2D plane.

For example, in Figure 2.1, when the round approximation is applied to calculate node's local density, only the density of node C will increase by one because the pattern P is nearest to node C other than node A, B and D. So the local density of A, B, C and D will increase by $\Delta\, density_A = 0$, $\Delta\, density_B = 0$, $\Delta\, density_C = 1$ and $\Delta\, density_D = 0$, respectively. Notice $\Delta\, density_A + \Delta\, density_B + \Delta\, density_C + \Delta\, density_D = 1$.

B. Soft Decision using Fuzzy Type Approximation

Suppose each pattern has $ndim$ dimensions. For each dimension d, the fuzzy type membership function (Figure 2.2) follows:

$$f_d(N_d) = \begin{cases} 1 - abs(P_d - N_d), & abs(\Delta_d) \le 1\ (cell\ length) \\ 0, & otherwise \end{cases} \tag{39}$$

where P_d is the coordinate of pattern in the d-th dimension, N_d is the coordinate of grid node in the d-th dimension and $\Delta_d = abs(P_d - N_d)$. A pattern P contributes different weights of densities to these grid nodes within distance of 1 in each dimension.

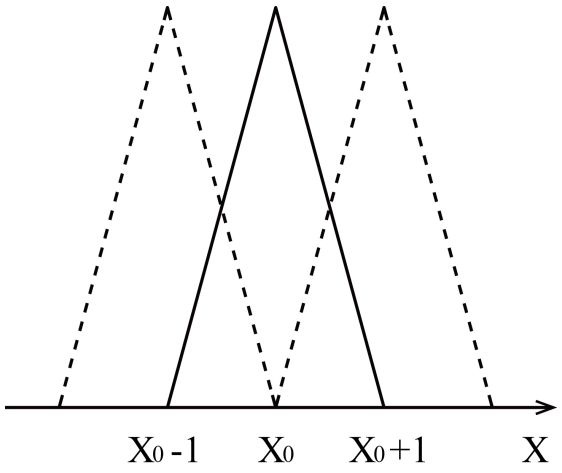

Figure 2.2 Fuzzy type membership function of soft decision.

When applying soft decision using fuzzy type function to calculate the node's local density, each pattern contributes to all the nearby nodes by different weights (equation (39), not only to the nearest.

For a node, $node\_k$, with coordinates $(N_1, N_2, \dots, N_{ndim})$, its node's local density is calculated as:

$$\rho_{node\_k} = \prod_{d=1}^{ndim} f_d(N_d) \tag{40}$$

The nodes' local densities are found by applying equation (40) to all the patterns.

Take the case in Figure 2.1 for instance, when the fuzzy type function is applied to calculate node's local density, the densities of all the four around nodes, A, B, C, D, will increase but by different values. The density of A, B, C and D will arise with

$$\begin{aligned}
\Delta\, density_A &= [1 - abs(2.6 - 2)] * [1 - abs(2.8 - 2)] = 0.08, \\
\Delta\, density_B &= [1 - abs(2.6 - 3)] * [1 - abs(2.8 - 2)] = 0.12, \\
\Delta\, density_C &= [1 - abs(2.6 - 3)] * [1 - abs(2.8 - 3)] = 0.48, \\
\Delta\, density_D &= [1 - abs(2.6 - 2)] * [1 - abs(2.8 - 3)] = 0.32,
\end{aligned}$$

respectively. Notice that the sum of density increments equals to 1, which is $\Delta\, density_A + \Delta\, density_B + \Delta\, density_C + \Delta\, density_D = 1$.

C. Spline Technique

For a better view of node's local density, spline technique [37] can be used. Specifically, the cubic spline interpolation technique is used between nodes in this chapter.

D. Discussion of Hard Decision, Soft Decision and Grid Size

Both hard decision and soft decision can be applied to compute node's local density. However, one of them will be preferable as the density error is concerned. To do the comparison, the square of $peaks$ function, or $peaks^2$, is set as the *target density function* (equation (41) and the surface of the target density function is called the *target density surface* (Figure 2.3).

$$z = [3 * (1 - x).\hat{}2.* exp(-(x.\hat{}2) - (y + 1).\hat{}2) \ldots$$
$$- 10 * (x/5 - x.\hat{}3 - y.\hat{}5).* exp(-x.\hat{}2 - y.\hat{}2) \ldots$$
$$- 1/3 * exp(-(x + 1).\hat{}2 - y.\hat{}2)].\hat{}2 \quad (41)$$

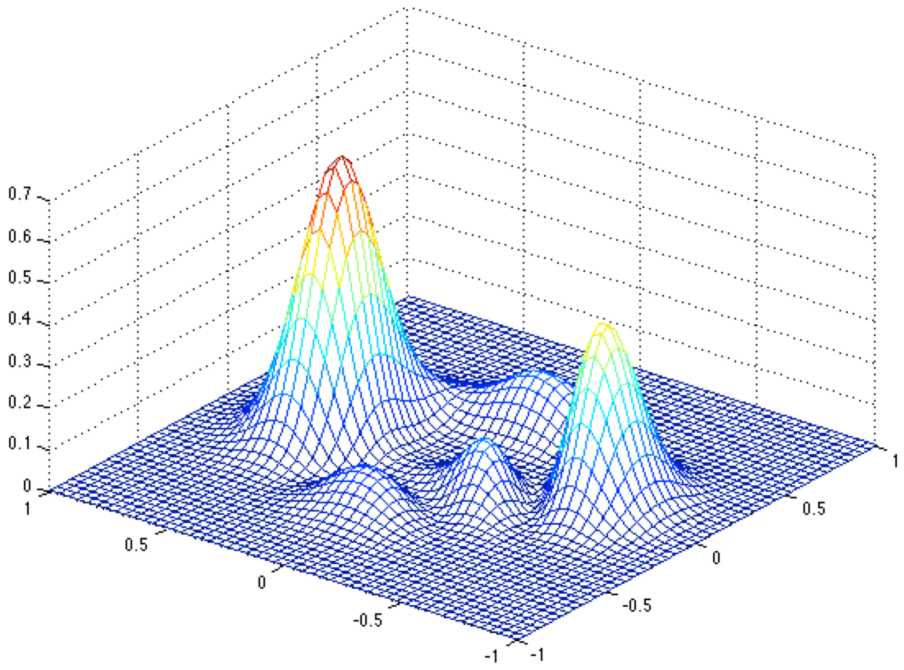

Figure 2.3 Target density surface: $peaks^2$. It has five different density distributions, shown as five peaks in this figure.

The density error is the difference between the *target density surface* and the *output density surface*. Figure 2.4 shows error comparisons using hard and soft decisions for several cases with different numbers of patterns. Please notice that soft approach always produces smaller errors but hard decision method is about 30% less computationally intensive.

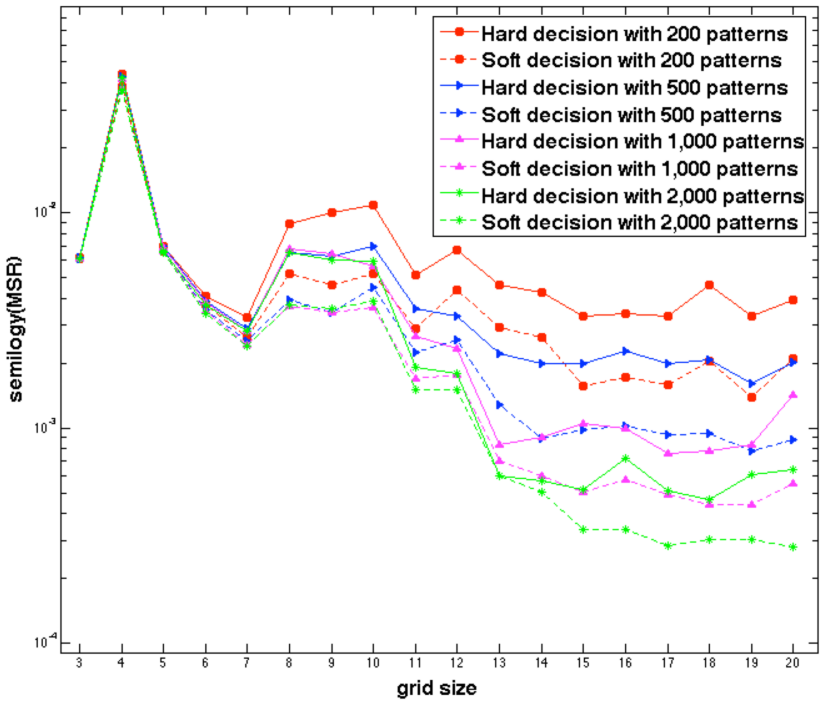

Figure 2.4 Density error comparisons using hard decision (solid lines) and soft decision (dotted lines). Notice that the soft decision strategy (dotted lines) achieves smaller density mean squared error (MSE) than the hard decision strategy (solid lines with the same color).

Spline technique can be used in the output density surface to get a more smooth surface with more nodes. As the Figure 2.4 shows, if the grid size is too small, i.e. 3, there will be huge errors in density surface. When the grid size becomes larger, the density error will go down. When the grid size is larger than 13 in this instance, the density error will change slowly. That means that too small grid size may fail to do this job, and too large grid size is not effective in computing nodes' densities. Four different numbers of patterns are applied, and all lead to the same result that there will be smaller density errors using soft decision than hard decision.

2.3.3 Generate the Decision Graph

A. Calculating the Minimum Distance δ_i between Node i and Any Other Node with Higher Node's Density

The minimum distances δ_i can be computed equation (7) described in Section 1.2.4.

Here, a constant variable of 2 is added to the node with the maximum local density, making the node prominent in the decision graph.

B. Sparse Matrix Operations

However, even though this presented grid-based method has largely decreased the number of calculations by generating small size of the grid in each dimension, it is somehow not so perfect in computation complexity, especially when the dimensions of patterns are extremely huge. At the same time, it is noted that most parts of the nodes in the grid we generated have no densities because of the blank margin, which is used to separate different clusters, between each cluster. That means, actually, there is no need to deal with these zero-density nodes that will surely cost time. Thus, instead of computing all the nodes of the grid, sparse matrix that keeps the non-zero nodes will be used and processed in this proposed method. In this way, relatively dense grids can be used for multidimensional cases.

C. Generate the Decision Graph

Aiming to choose the patterns with large local density ρ_i and large minimum distance δ_i as the cluster centers, the decision graph is generated with x-coordinate ρ_i and y-coordinate δ_i. For each pattern i, now its local density ρ_i and the minimum distance δ_i are known. In FSFDP [14], it shows another way to select the cluster centers is by plotting the figure with x-coordinate i and y-coordinate $\gamma_i = \rho_i * \delta_i$ and then choosing the patterns with larger γ_i as cluster centers. After selected the cluster centers in the decision graph, patterns will be assigned into different clusters based on the δs.

After selected the nodes, or cluster centers, in the decision graph, nodes will be assigned into different clusters based on the minimum distances δs. Node will be categorized as the same

cluster with its nearest node with a larger density. After all non-zero-density nodes are clustered, the patterns will be classified as the same cluster with its nearest node, which is already clustered.

2.3.4 A Simple Example Illustration

As the clustering of the *2spiral* dataset [38] is a big challenge for conventional centroid-based clustering methods, e.g. K-means [1] in Figure 2.5, the *2spiral* benchmark generated using Matlab built-in *twospiral* function is used to illustrate how the presented clustering method works (Figure 2.6-Figure 2.10).

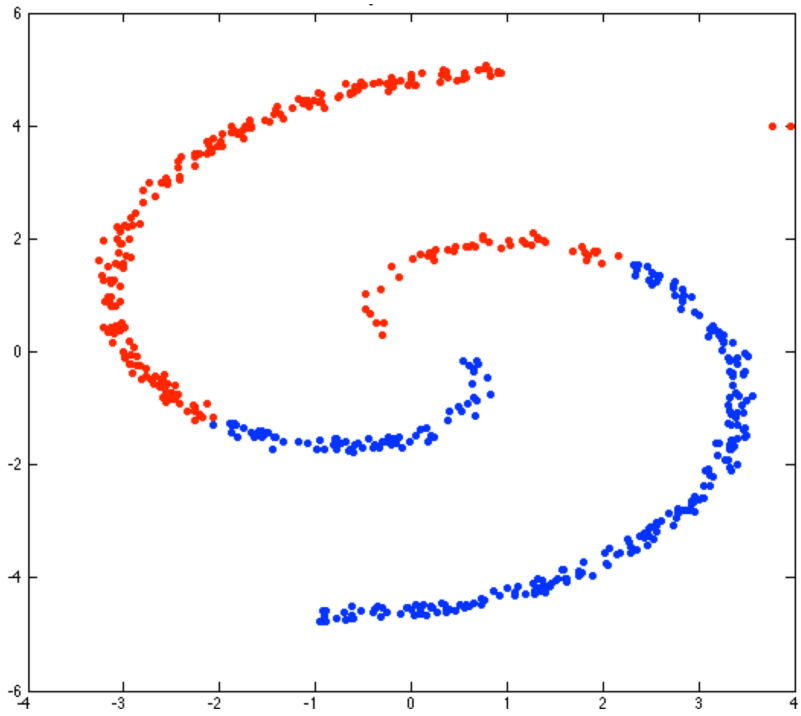

Figure 2.5 Clustering result of *2spiral* data using K-means method. Notice that K-means method is not efficient for clustering datasets with complex shapes and may cause huge misclassification.

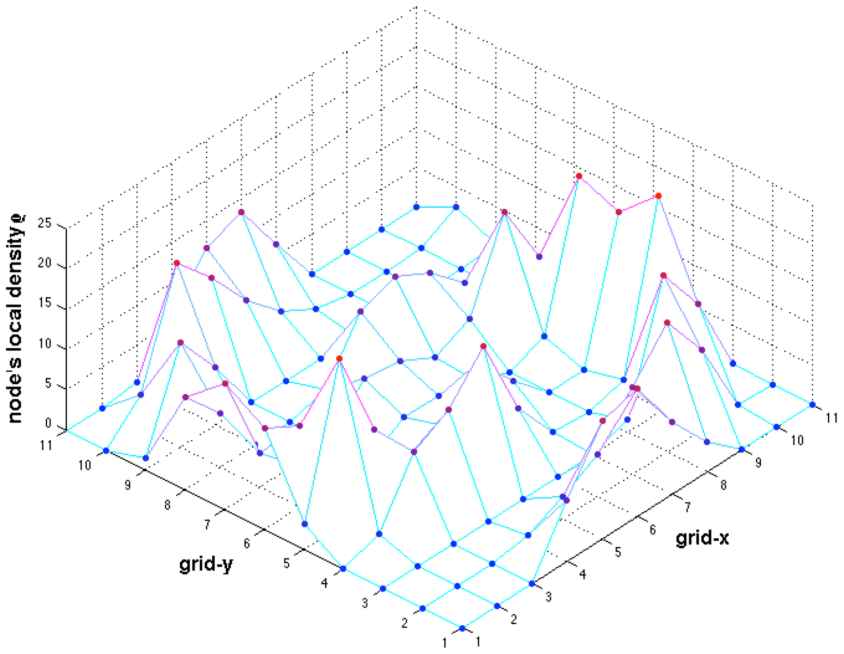

Figure 2.6 Node's local density (z-axis) using soft decision shown in a [1,11] range grid.

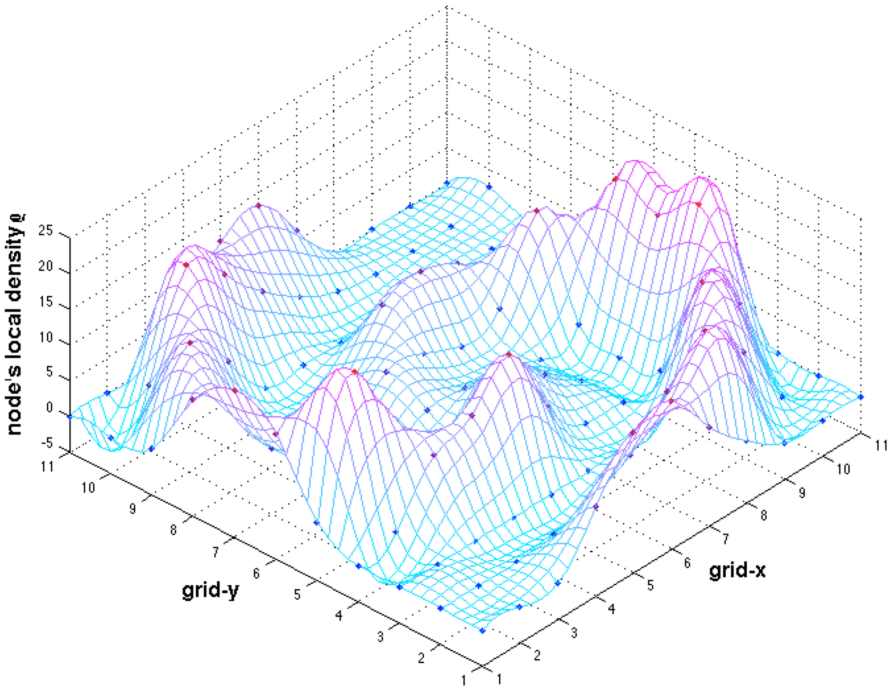

Figure 2.7 Node's local density smoothed with a spline routine on Figure 2.6.

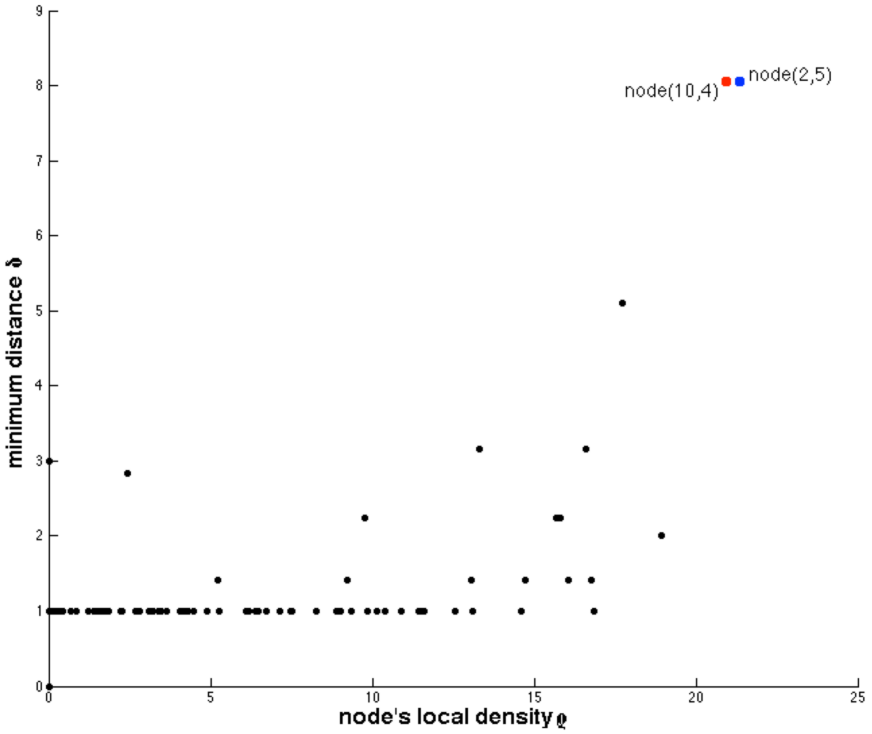

Figure 2.8 Decision graph for the *2spiral* case.

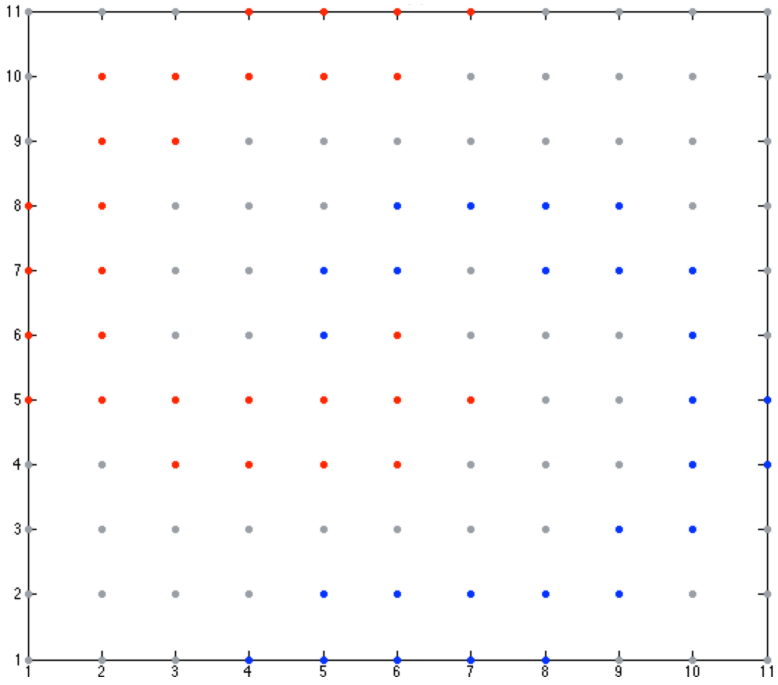

Figure 2.9 The clustering result of grid nodes associated with patterns for the *2spiral* case. Gray node stands for noise around it.

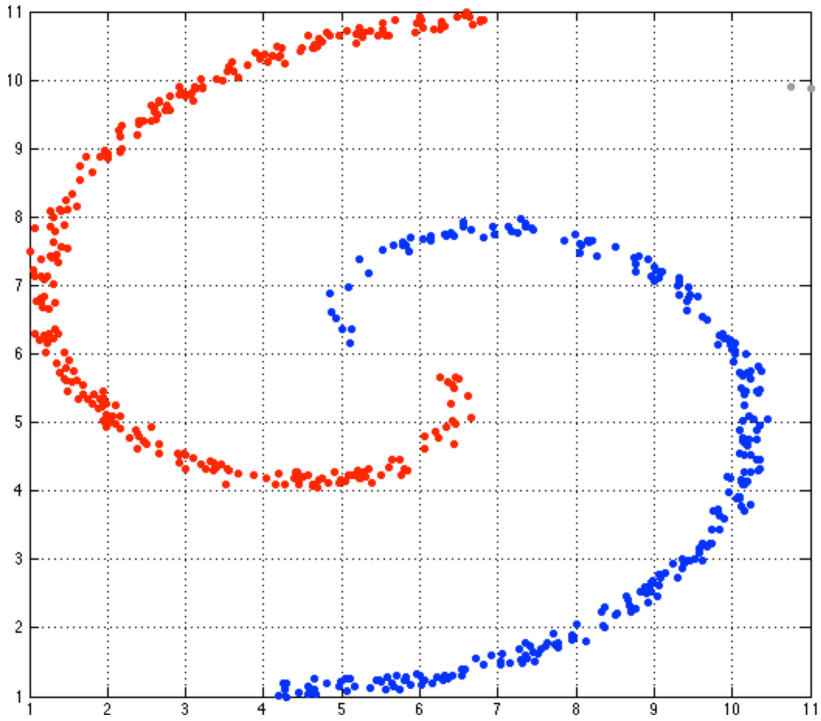

Figure 2.10 Classified *2spiral* patterns using the presented clustering method. Gray dots are the noise. Identified results can be obtained as well using the FSFDP. Notice that the FSFDP requires about 72 seconds to process this *2spiral* dataset while the presented algorithm obtains the same result with about 0.01 seconds. Same clustering result will be achieved using hard decision.

As the Figure 2.10 shows, the presented clustering method succeeds to categorize the *2spiral* data into two clusters, and if a cutoff density threshold is set to detect noise in the original dataset, the proposed clustering method is capable of detecting noise (gray dots in Figure 2.10). However, the commonly used K-means method misclassifies these patterns (Figure 2.5). The node's local density is clearly visualized in Figure 2.6, with the use of spline technique, even a better view of the node's local density (Figure 2.7) is produced after estimating three more density values between neighbor nodes. In the density-based clustering methods, the center is defined as the data point with the largest local density, so the center should have a larger density (ρ) and a relatively large distance (δ). When the decision graph is generated, one can select the

node(s), or the center(s), in the upper-right part of the decision graph (Figure 2.8). After the selection of node(s)/center(s), the node(s)/center(s) will be color noted with the position(s) in the decision graph Figure 2.8, e.g. *node(10,4)* means the node with x-coordinate 10 and y-coordinate 4 in the standard grid. Then, the nodes with non-zero density will be classified (Figure 2.9) based on the node(s)/center(s) you selected in the decision graph. At last, all patterns will be clustered (Figure 2.10) into the same category with the nearest node, which is already clustered. Experiments on several datasets are shown in the following section.

2.4 Experimental Results

Several datasets have been used to test the proposed Clustering algorithm based on Grid and Finding Density Peaks (CGFDP). These data sets have a different number of clusters with different shapes and a different number of patterns. All the experiments are implemented based on the same software and hardware: Matlab R2013a in OS X operating system with Intel Core i5 (I5-4258U) @2.4GHz 8.00GB memory.

2.4.1 fig2_panelC Dataset

This data set is provided in the FSFDP. It has 1,000 patterns generated from a probability distribution with nonspherical and strongly overlapping peaks.

2.4.2 fig2_panelB Dataset

This is a dataset generated from the same distribution with the fig2_panelC dataset, but with a larger pattern number of 4,000. It will cost about 0.13 seconds to get a final clustering result when the presented method is applied. As a contrast, it runs about 15 hours when using the FSFDP method.

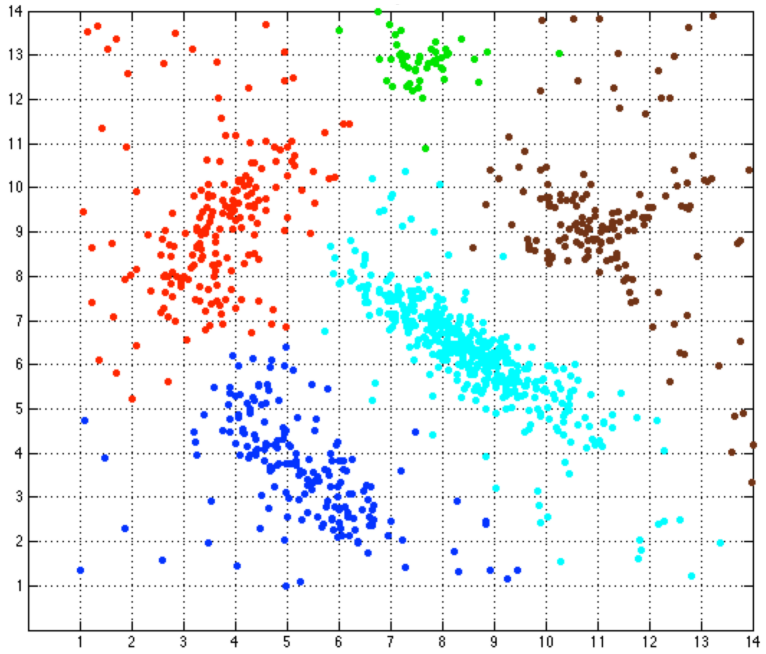

Figure 2.11 Clustering result of fig2_panelC data using the proposed algorithm with hard decision.

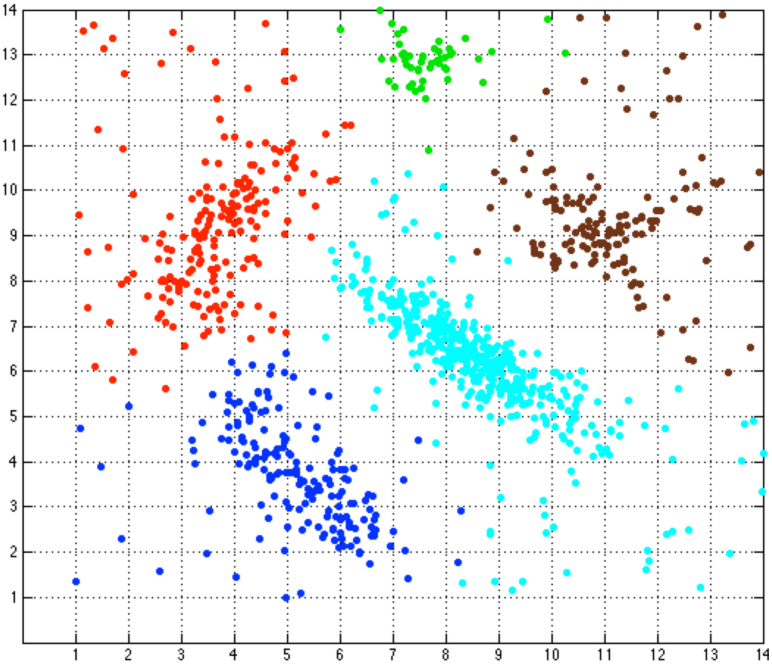

Figure 2.12 Clustering result of fig2_panelC data using the proposed algorithm with soft decision.

2.4.3 FLAME Dataset

FLAME data [39] comes from the problem for clustering DNA microarray data.

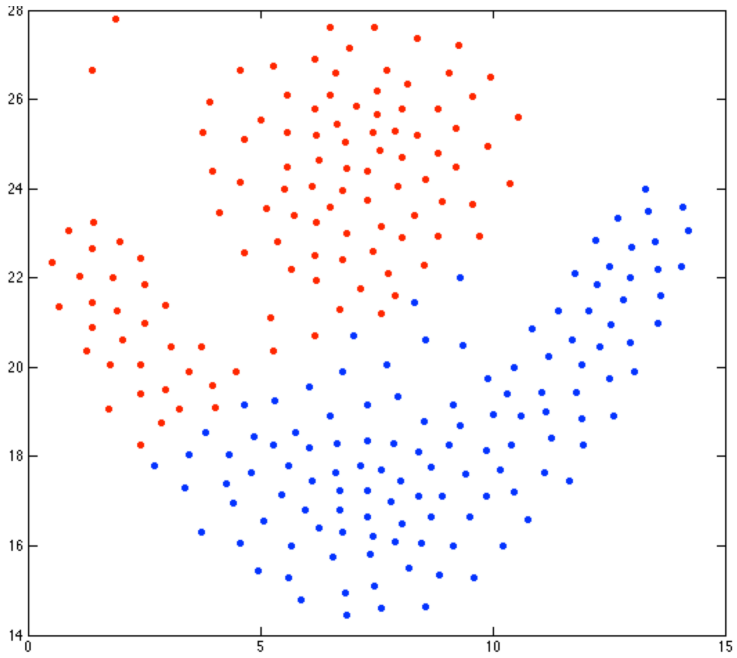

Figure 2.13 Classified FLAME patterns using K-means method. Notice that the K-means method is not capable of clustering patterns with unregular shapes with less misclassification.

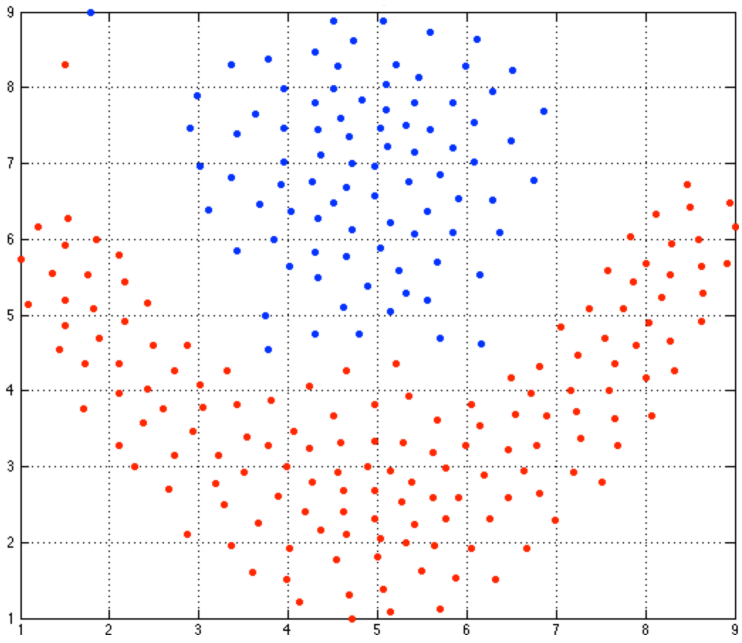

Figure 2.14 Clustering result of FLAME data using the proposed algorithm with hard decision.

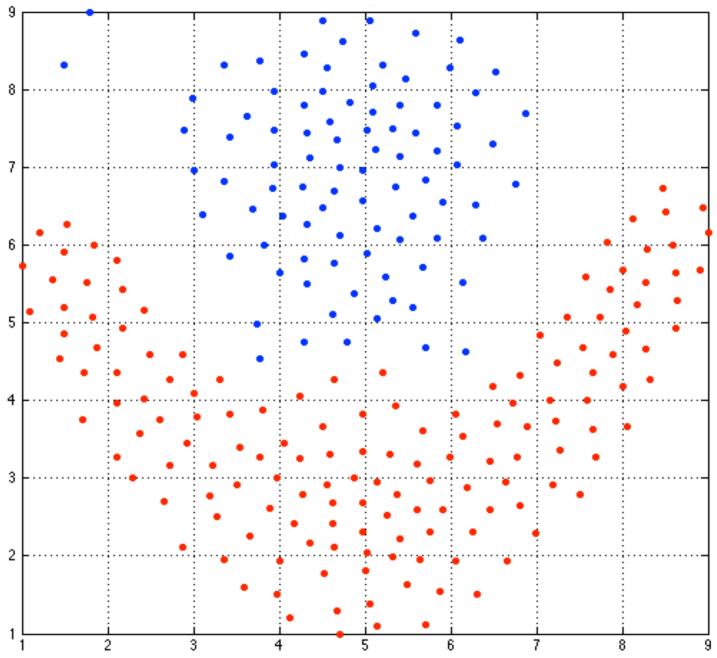

Figure 2.15 Clustering result of FLAME data using the proposed algorithm with soft decision.

2.4.4 S3 Dataset

The S3 data [40] for clustering purpose has 5,000 patterns with 15 clusters.

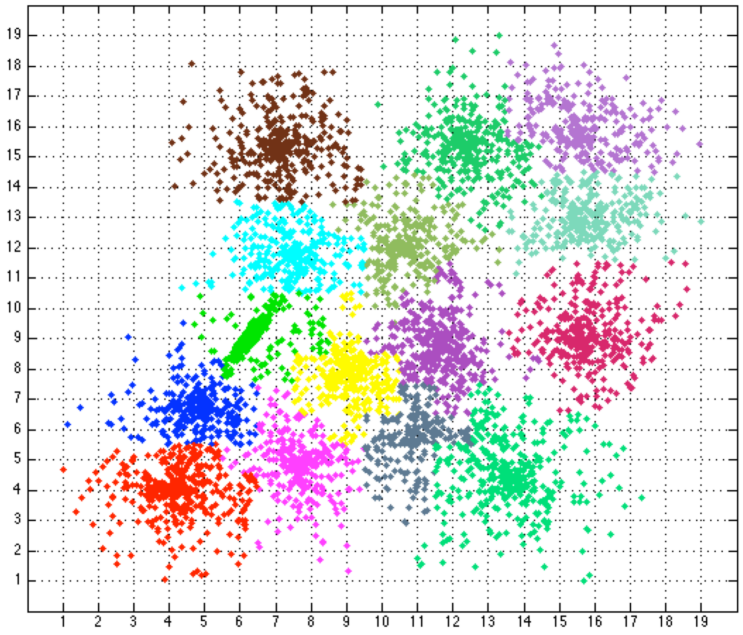

Figure 2.16 Clustering result of S3 data using the proposed algorithm with hard decision.

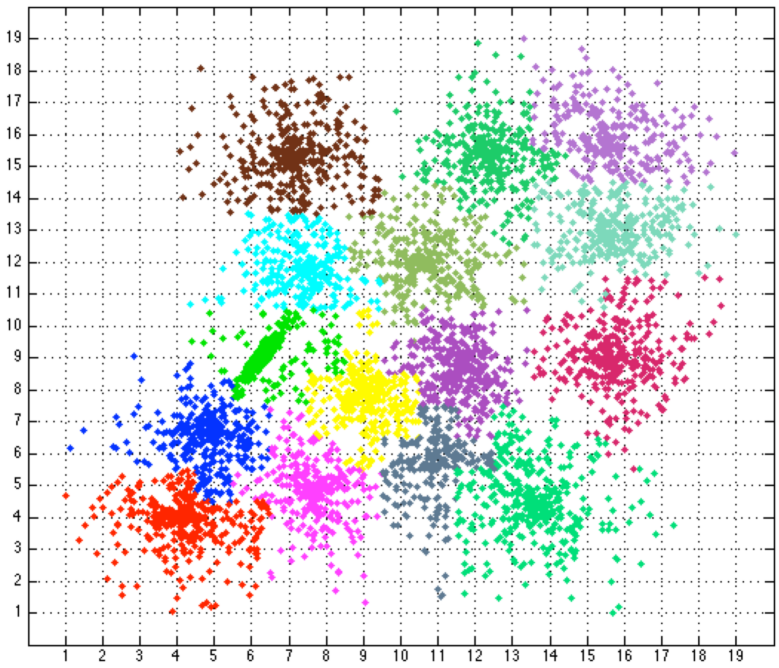

Figure 2.17 Clustering result of S3 data using the proposed algorithm with soft decision.

2.4.5 Aggregation Dataset

Aggregation [41] is a dataset with no noise in it, also tested in FSFDP method. The clustering results are not shown here, as most of the clusters are ellipse shape and don't have too many numbers of patterns, so it's not hard to cluster it.

2.4.6 Results Analysis

Totally six datasets (including the *2spiral* benchmark) are applied to test the presented CGFDP algorithm. Throughout the experiments, the proposed CGFDP method using hard decision or soft decision can give us satisfying clustering results, while the K-means clustering method fails to cluster the FLAME dataset (Figure 2.13). The difference in the clustering results between hard decision and soft decision happens only in the sparse nodes/patterns and the boundary between clusters, where it may cause classification errors. If we treat the sparse

nodes/patterns and the boundary between clusters as noise by defining a threshold for node densities as noise, the proposed method is possible to categorize data sets with noise.

Generally, a larger size of grid is more capable but will cost more time, on the other hand, a standard grid with a too small size, e.g. 3, may fail to do this job. Usually, a standard grid with the size of less than 20 in each dimension is capable of clustering a given dataset.

Our method does not require to compute the mutual distances between patterns or know the multidimensional density function. It requires computation time even 10,000 shorter for cases with different numbers of patterns (Table 2-1). Therefore, it outperforms the FSFDP and even other conventional approaches regarding efficiency and effectiveness. Even we increase the grid size within a reasonable region, the processing time is not changing dramatically. The computation time in processing these six experimental datasets with different grid sizes is shown in Figure 2.18 with the increase of grid size.

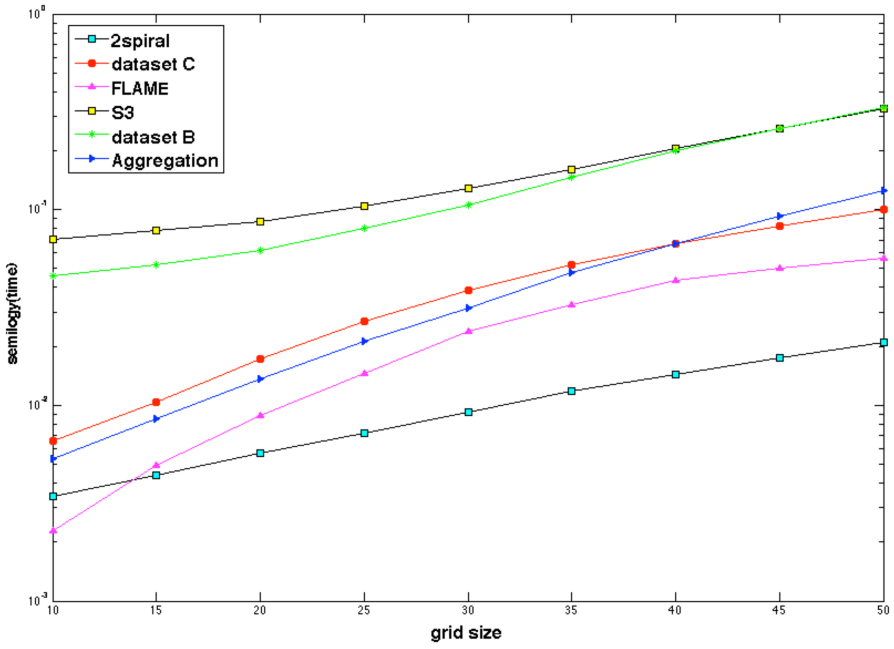

Figure 2.18 Computation time (in seconds) comparisons for six experimental datasets with different grid size.

The presented CGFDP method gives basically the same results as obtained in the FSFDP so the results of clustering for *PanelB* and *Aggregation* benchmarks used in [14] are not shown here but the comparison of computation time is included in Table 2-1. It lists the time comparisons between the averaging processing time of the presented CGFDP method with different grid sizes and the FSFDP, which shows that the presented method is much faster than the FSFDP.

Table 2-1 Comparisons of processing time on six benchmarks

Benchmarks	FSFDP	CGFDP (hard decision)	CGFDP (soft decision)
2spiral (n=537;ndim=2)	71.1021 sec	0.0046 sec	0.0109 sec
PanelC (n=1,000;ndim=2)	1.1373e+03 sec	0.0211 sec	0.0447 sec
FLAME (n=240;ndim=2)	1.5482 sec	0.0070 sec	0.0264 sec
S3 (n=5,000;ndim=2)	8.0236e+04 sec	0.1748 sec	0.1586 sec
PanelB (n=4,000;ndim=2)	5.2782e+04 sec	0.0729 sec	0.1425 sec
Aggregation (n=788;ndim=2)	414.4445 sec	0.0193 sec	0.0425 sec

2.5 Conclusion

This chapter presented a grid-based clustering method by finding density peaks, which is fast and practical to classify data (even with noise) into different categories. It can easily scale up to cluster datasets with different sizes of dimensions. With the advantage of the friendly interactive interface in the density-based method by finding density peaks, at the mean time, it decreases enormously computation complexity to $O(n * grid\_size)$. The proposed algorithm follows three steps: (I) normalizing and expanding the original data set into a standard grid; (II) calculating the node local density; (III) generating the decision graph. After the selection of centers in the decision graph, the individual pattern will be assigned to the same cluster with its nearest node that is already categorized. During the second step, either hard decision or soft decision can be applied. In terms of efficiency and effectiveness, experiments on real-world

datasets show that the proposed method significantly outperforms the FSFDP (see Table 2-1) in the process of calculating the local densities and assigning data points into different categories due to the use of the standard grid and sparse matrix technique respectively.

CHAPTER 3 A FAST DENSITY AND GRID BASED CLUSTERING METHOD FOR DATA WITH ARBITRARY SHAPES AND NOISE

As mentioned in CHAPTER 2, the grid based clustering can accomplish the tasks very fast. In this chapter, a density and grid based (DGB) clustering method for data with arbitrary shapes and noise is described. As most of the conventional clustering approaches work only with round-shape clusters, it needs to explore other methods for proceeding classification for clusters with arbitrary shapes. Clustering approach by fast search and find of density peaks (FSFDP) and density based spatial clustering of applications with noise (DBSCAN), and so many others are reported to be capable of completing this task but limited by its computation time of mutual distances between points or patterns. Without the calculation of mutual distances, this work presents an alternative method to fulfill clustering of data with any shape and noise even faster and more efficient. It was successfully verified in clustering industrial data (e.g. DNA microarray data) and several benchmark datasets with different kinds of noise. It turned out that the proposed DGB clustering method is more efficient and faster in clustering datasets with any shape than the conventional methods.

3.1 Introduction

An essential routine to pre-proceeding a given industrial data is to seek its clustering structure. Many applications in the industrial area using various clustering methods can be found in the literature, e.g. [42][43] etc. Clustering approaches come along with different definitions of clusters. The expectation-maximization (EM) algorithm [3] categorizes patterns into the cluster with maximum likelihood. The assumption of EM clustering algorithm is that the cluster is a combination of patterns that have most likely the same distribution. The EM algorithm fulfills

this task by optimizing the distribution functions of clusters. Applications using EM are reported in [44][45]. The widely used K-means method [1] finds the clusters by iteratively computing the distances from patterns to the gravity centers of clusters until converge. It assumes that the patterns, which belong to the same cluster, are located around the cluster's gravity center. Various applications based on K-means method can be seen in [46][47]. Another alternative approach is called hierarchical clustering [2] method, which keeps the property that patterns with small distance are more related than with large distance.

The idea of grid-based clustering was proposed almost twenty years ago, along with many publications. The GRIDCLUS method [48] applies grid technique to implement the hierarchical clustering approach. The patterns are grouped into blocks and clustered with respect to the blocks by a topological neighbor search algorithm. It is reported the GRIDCLUS method is much faster than the traditional hierarchical clustering approach.

Wei Wang et al. proposed a STatistical INformation Grid-based clustering method (STING) [49] to cluster spatial databases. The spatial area is divided into rectangle cells at different levels of resolution, which forms a hierarchical structure. Let the root of the hierarchy be at the 1st level, its children at level 2, etc. The number of layers could be obtained by changing the number of cells that form a higher-level cell. A cell in level i corresponds to the union of the areas of its children in level $i + 1$. In STING, each cell has 4 children and each child corresponds to one quadrant of the parent cell. Statistical information of each cell is calculated and stored beforehand and is used to answer spatial data mining queries. For each query, it starts at the root and proceeds to the next lower level using the STING index. Then, it requires computing the likelihood that this cell is relevant to the query at some confidence level using the parameters of this cell. Only children of likely relevant cells are recursively explored. Repeat this process until

the bottom level is reached. The STING algorithm stops with all the requirements of queries are met. Only two-dimensional spatial space is considered in this algorithm.

The idea of grid in clustering algorithms is even more useful and powerful today. In the GRIDCLUS method, the grid is divided into rectangular but not limited to rectangular segments with different sizes, which conceals the overwhelming capability of the universal grid with universal size. Although an uniform size of rectangular in each layer is involved in the STING algorithm, the strategy of answering queries may lead inaccuracy because its probabilistic nature may indicate a loss of accuracy in query processing. However, one outstanding feature when using the grid-based approaches is the less consuming time. For example, the time complexity of STING is $O(n)$. In order to improve the processing accuracy, density-based approaches and grid-based approaches can be combined to achieve that.

In density-based clustering methods, the cluster is categorized with a higher density of patterns in it than out of it. In density based spatial clustering method of applications with noise (DBSCAN) [13], it illustrates that DBSCAN can be applied to detect clusters with arbitrary shapes. One application using DBSCAN can be found in [50]. However, the computation time of calculating and sorting the mutual distances is numerous.

Another density-based clustering algorithm is called clustering method by fast search and find of density peaks (FSFDP) [14] published recently in Science. It uses the pattern with the largest local density as the cluster center, not the conventional gravity center so that it can be applied to categorize the data sets with irregular shapes but not all of them. However, it burdens the computation complexity when generating the distance matrix, which requires the calculation of mutual distances between patterns.

In this chapter, we present an attempt to cluster datasets with arbitrary shapes and noise. Instead of calculating Euclidean distances between mutual patterns, only distances between a much smaller number of grid nodes are needed. Moreover, a new algorithm of computing densities was proposed, where a pattern and its surrounding nodes are involved. It turns out to be faster and feasible to categorize patterns with arbitrary shapes and irregular noises.

3.2 Density and Grid Based Clustering for Data with Arbitrary Shapes

3.2.1 Mountain Ridge

It comes from the natural phenomenon that geographers identify and cluster different mountains by drawing and analyzing their ridges. As long as the mountain ridges are detected, it becomes easy to cluster and categorize mountains. An example of natural mountain ridge of Big Savage Mountain (MD and PA, USA) [51] is shown in Figure 3.1, the Big Savage Mountain. From the view of mountain ridges, we can clearly see how each mountain goes, no matter big or small. In a similar way, can we view data as several mountain ridges? One way to find out the data ridges is by counting data densities as their mountain heights. Data point with the largest local density is the mountain peak and data point with the smallest local density is the mountain bottom. Other data points are among peaks and bottoms. An example of showing what the mountain ridges of a dataset looks like can be found in Figure 3.2 and Figure 3.3.

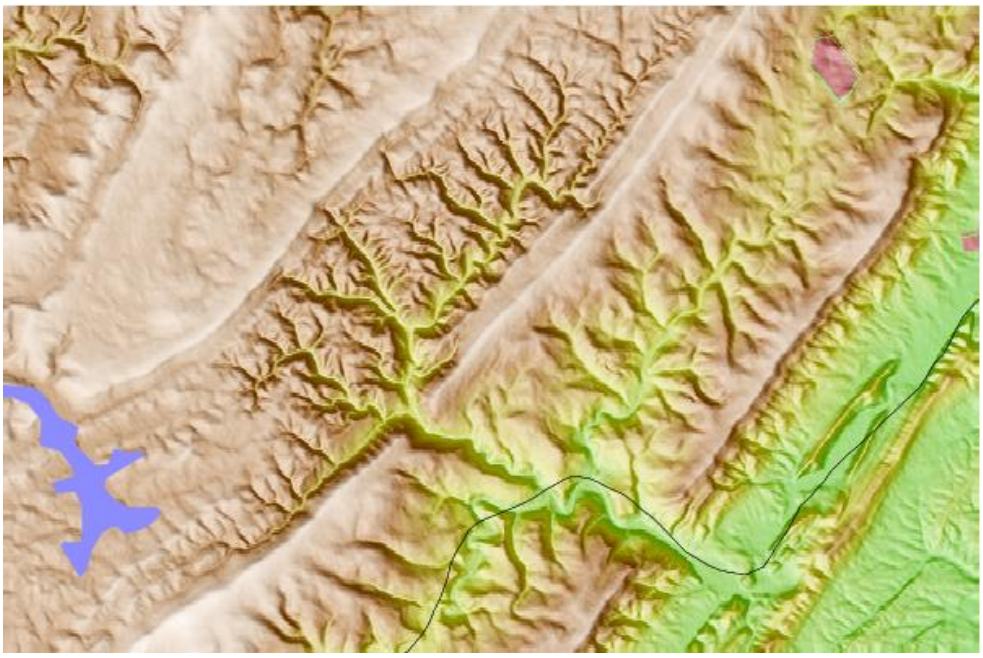

Figure 3.1 The Big Savage Mountain (MD and PA, USA) ridges.

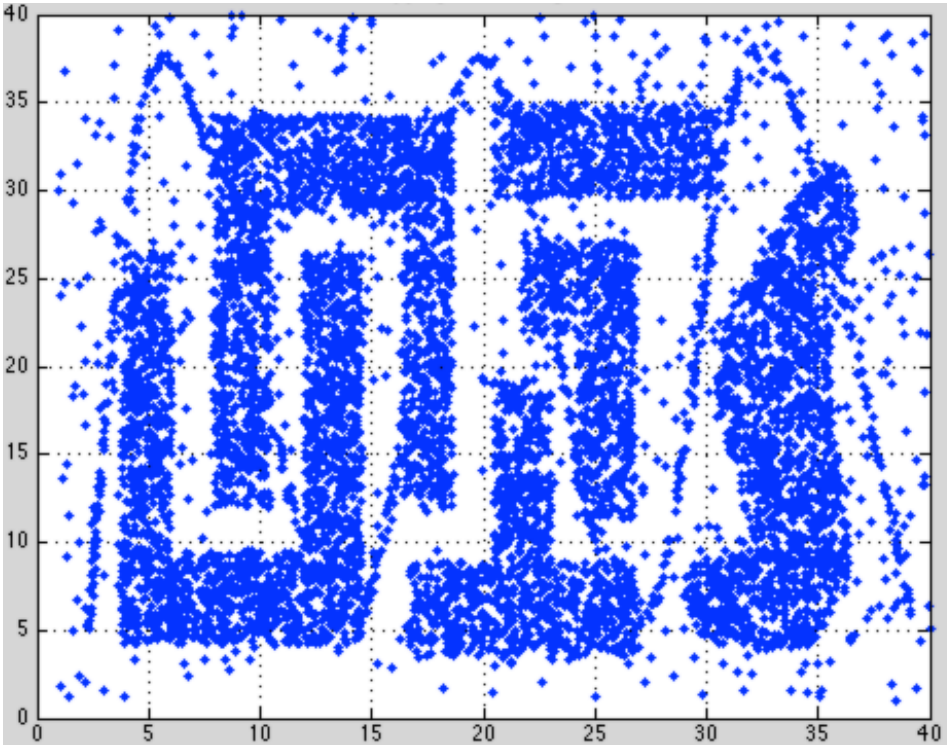

Figure 3.2 A data instance for mountain ridge explanation.

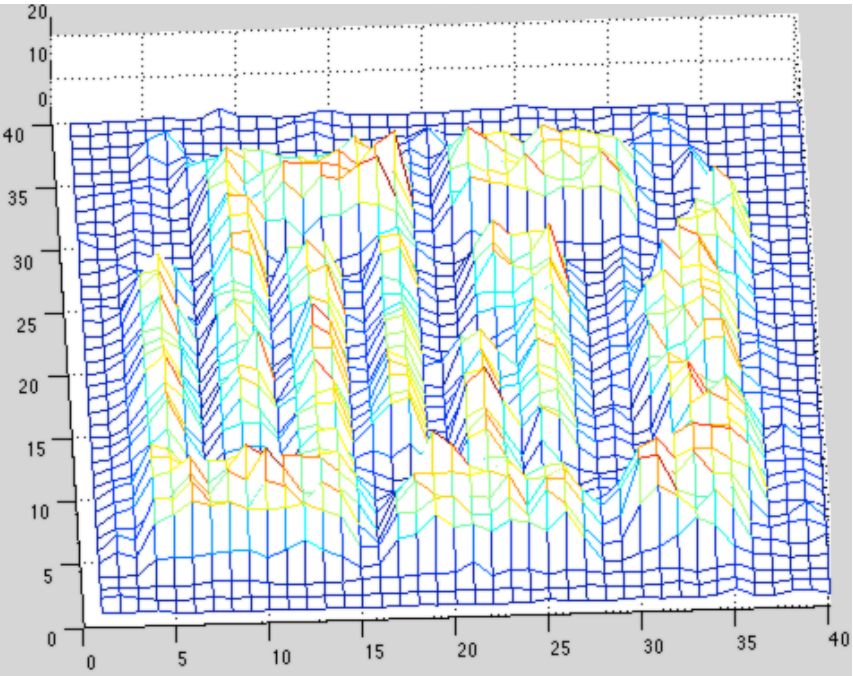

Figure 3.3 Densities of data in Figure 3.2. These data densities form shapes of mountain ridges.

3.2.2 Finding Data Mountain Ridge

As the outlook of nodes' local densities is like mountains with different heights, so the task of clustering is redefined as finding the mountain ridges. To fulfill it, mountain ridges are detected one by one starting with the peaks, which are the nodes with higher local densities. For the first mountain ridge, it starts with the grid node with the largest local density among all nodes. Then it labels the neighbor nodes, which have distances of 1, into this mountain if the neighbor nodes are not the edges. The mountain edges are the nodes with densities smaller than $Edg\%$ of this mountain peak density. From the merged node(s), keep merging its/their neighbor node(s) into this mountain until all edges of this mountain are reached. Then it continues to find the next mountain ridge, starting with the grid node that has the largest node's local density among the unlabeled nodes, and keep labeling nodes until all mountain ridges are detected. There are two strategies to terminate mountain ridges searching. One of them is by checking if the starting node

has a larger local density than the node containing noise or not. Another way to terminate it is to see whether the number of nodes labeled into one mountain ridge is larger than some constant or not. The number of mountain ridges corresponds to the number of clusters, which is revealed automatically through this procedure.

Basically, mountain ridges give a view of the outlooks of clusters if noise exists in the data. In other cases with no noise in the data, e.g. *3spiral* dataset in Figure 3.7, each mountain ridge contains all the nodes that belong to one cluster. After the non-noise nodes are labeled as different mountain ridges, the patterns will be classified as the same mountain, or cluster, as its nearest node that is already labeled. The classification of border patterns and the detection of noise, if any, are described in the following section.

3.2.3 Mountain Border and Noise Detection

If there is no noise in the data, after the completion of labeling nodes into different mountain ridges, individual patterns will be classified into their nearest already-labeled nodes. For the case of data with noise, the rest unclassified sparse patterns could be either borders or noises.

By setting the $Edg\%$ factor, the mountain edges, or the border nodes, can be easily detected when finding the mountain ridge. For a node containing noise, its local density is small, even smaller than the $Edg\%$ factor. By setting a $Noise\_cut\_thre$ parameter, it's easy to filter the nodes containing noise. Usually, the value for $Noise\_cut\_thre$ is around 3 for white noise because there are commonly less than 4 noise points in one cell in the grid for 2-dimensional data case. Generally, the density based clustering method has the capability of detecting noise points with densities that are much smaller than the data points, no matter white or non-white noise. In the presented DGB method, points/patterns with sparse densities will be classified as

noise. However, some non-white noise points with densities close to patterns in clusters may be challenged for conventional clustering methods, e.g. DBSCAN and FSFDP, to detect them but not for the presented DGB clustering method. For example in the clustering result using DBSCAN of Figure 3.10, the strip-shaped/sin-shaped noise is misclassified as patterns instead of excluding it as noise between the clusters colored in red and blue.

Generally, the mountain ridges give a view of the outlooks of clusters. Therefore, it may need an extra procedure because there may exist some amount of patterns left unlabeled around border nodes. To be specific, there are basically two cases that border patterns will be classified into their nearby already-labeled nodes. Otherwise, they will be neglected as noise. Let's take the grid *cell* shown in Figure 2.1 for instance. For possible border patterns, at least one of the four neighbor nodes in the cell should have been labeled during the process of finding mountain ridges. In the case of only one of four surrounding nodes in the cell has been labeled, an individual pattern will be categorized into the cluster of the labeled border node with two satisfactions of (1) this pattern's nearest node is the labeled border node; (2) the contributed density to this border node from all the patterns inside this cell is larger than $Noise\_cut\_thre$. The contributed density to one node from its nearby cell is calculated during the step of computing node's local density. When two or three of the four surrounding nodes in the cell have been labeled, an individual pattern will be classified into the cluster of the labeled node, as long as the contributed density to these two or three nodes from all the patterns inside this cell is larger than $Noise\_cut\_thre$. Because enough-size grid is applied to make sure there will be grid cells as margins between clusters, these two or three labeled nodes will always belong to a same cluster. In conclusion, the sparse individual patterns will be processed as either border patterns or noise.

3.3 Density and Grid Based Clustering for Data with Noise

The procedure of the presented DGB clustering method are shown as follows.

1: Normalize and scale the data into $[1, N\_grid]$ grid; (see Section 2.3.1)

2: Calculate node's local density using soft decision; (see Section 2.3.2)

3: Find mountain ridges; (see Section 3.2.2)

4: If there is no noise in the data, label the individual patterns and the possible border patterns into different mountain ridges; otherwise, besides the labeling process, the detection of noise is also applied; (see Section 3.2.3)

5: End.

As the clustering of the *3spiral* dataset [52] is a big challenge for conventional centroid-based clustering methods, i.e. K-means (Figure 3.4), the *3spiral* dataset is used to illustrate how the presented clustering method works (Figure 3.5, Figure 3.6, Figure 3.7).

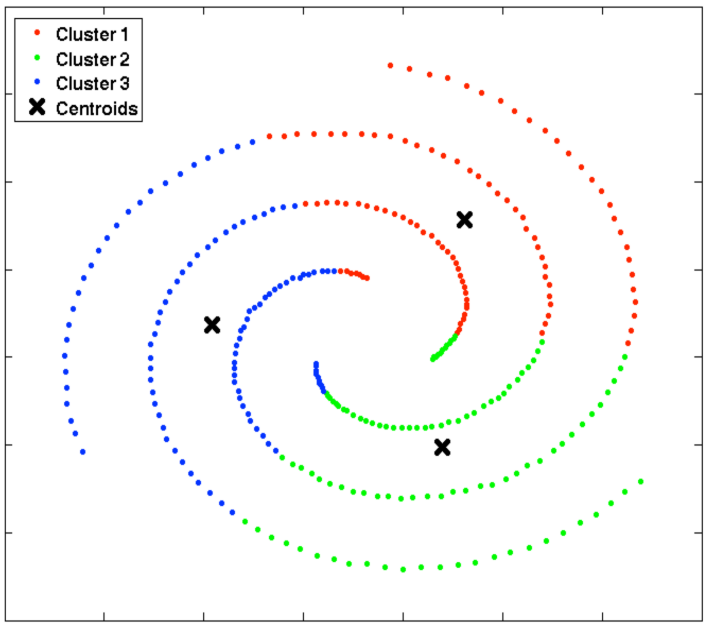

Figure 3.4 Clustering result of *3spiral* dataset using K-means method. Notice that K-means method is not efficient for clustering datasets with complex shapes and may cause huge misclassification.

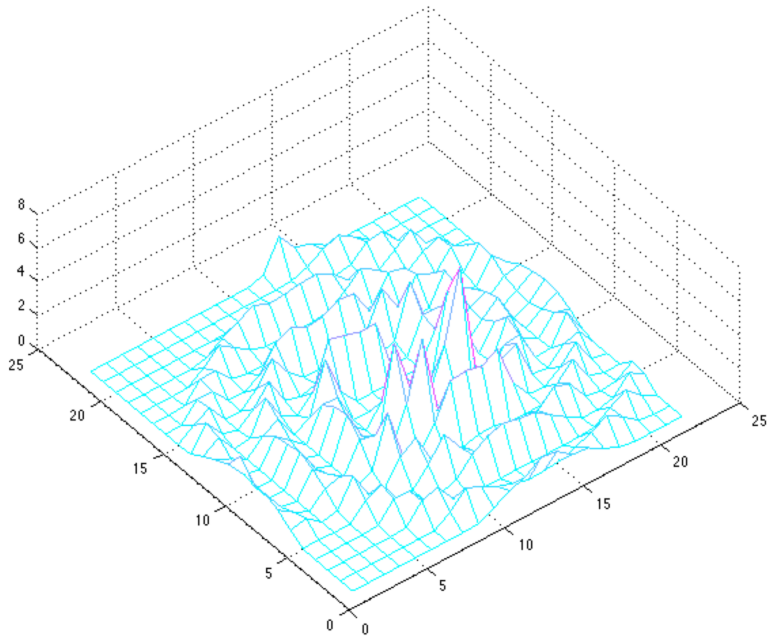

Figure 3.5 Nodes' local densities (z-axis) shown at grid nodes in a [1, 23] range grid.

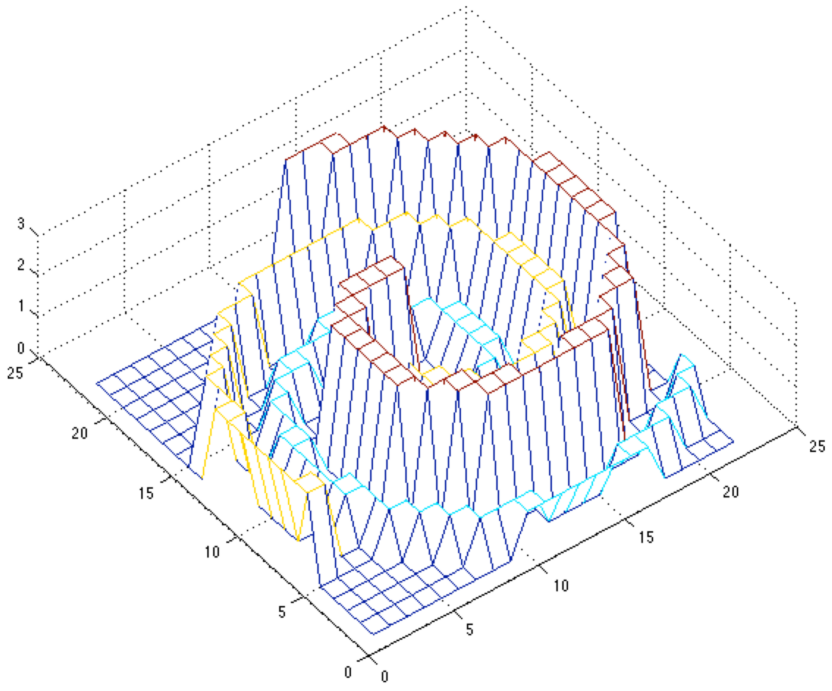

Figure 3.6 The mountain ridges found automatically for the *3spiral* case. Different clusters have different heights.

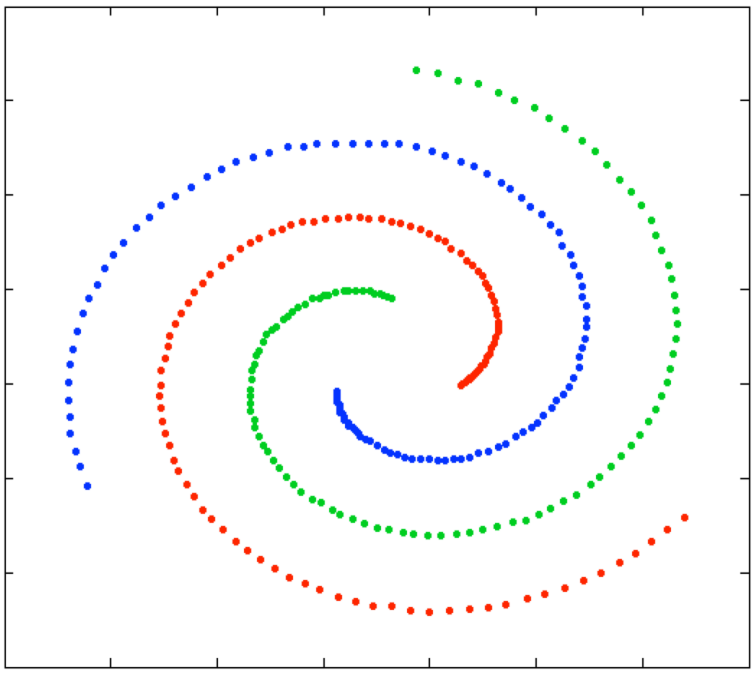

Figure 3.7 Classified *3spiral* patterns using the presented method. Identified results can be obtained as well using the FSFDP and DBSCAN. Notice that the FSFDP and original DBSCAN require about 1.7869 seconds to process this 3spiral dataset while the presented method obtains the same result with about 0.2 seconds.

3.4 Experimental Results

One real life medical industrial dataset, called FLAME [39], from gene expression profiles obtained with DNA microarrays is applied to the clustering process. Two additional chameleon datasets [53] with various shapes and noise are used to test the method as well. These datasets have different numbers of patterns and clusters. All the experiments are implemented based on the same software and hardware: Matlab R2013a in OS X operating system with Intel Core i5 (I5-4258U) @2.4GHz 8.00GB memory.

3.4.1 DNA Microarray Dataset - FLAME

Global DNA testing market is anticipated to reach USD 10.04 billion by 2022, according to a new study by Grand View Research (US) Inc. Primary revenue generating activity in the

market revolves DNA profiling obtained with DNA microarrays. Before further processing, it's necessary to conduct clustering analysis on DNA microarray data. Figure 3.8 shows the clustering result of DNA microarray dataset, *FLAME*.

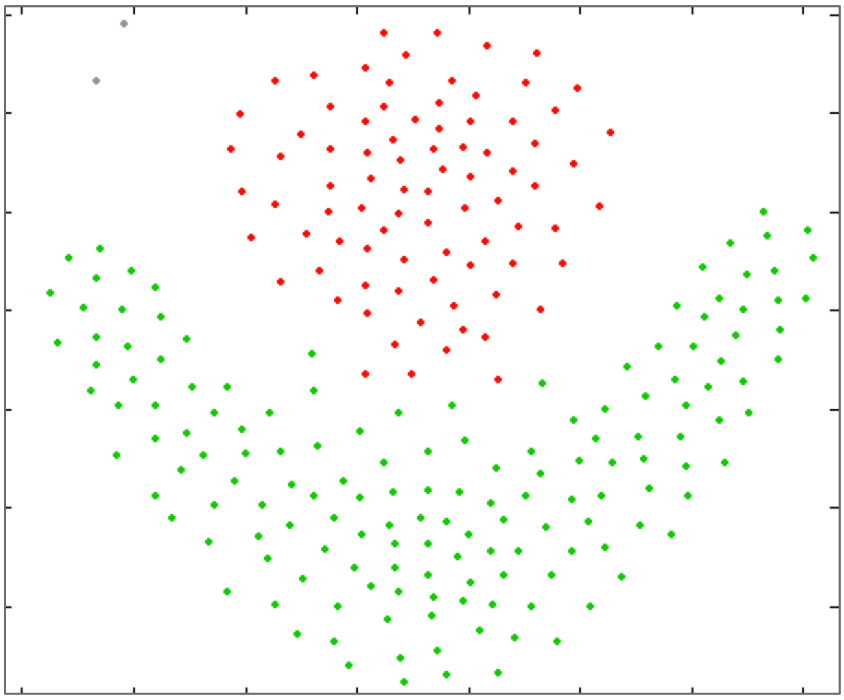

Figure 3.8 The clustering result of DNA microarray dataset FLAME using the presented clustering DGB clustering method. There are two clusters, and the outliers (gray dots in the upper left corner) are detected.

3.4.2 Case t4.8k Dataset and Case t8.8k Dataset

Datasets of t4.8k and t8.8k are two chameleon datasets, which have complex shapes and different noise. As the experimental figures shown, the presented DGB clustering method is capable of classifying datasets with complicated shapes, and even with non-white noises, e.g. sin-function-shaped noise in Figure 3.11. It gives basically the same results as obtained using DBSCAN with slightly difference, which is not significantly visible. There are some sparse patterns misclassified as clusters instead of noises between two neighbor clusters in the resulted

figures using DBSCAN (Figure 3.10), which is caused by the definition of directly density-reachable (equation (3)). While in the resulted figures using the proposed method (Figure 3.11), it shows a very clearly separated margin between neighbor clusters with a careful treat of borders and noise.

Moreover, for datasets with complicated shapes, FSFDP (e.g. Figure 3.9 and Figure 3.12) gives infeasible results as the presented DGB method. One critical issue using DBSCAN is the loss of effectiveness when the data has clusters with densities of different levels. It's resulted from the compromise of setting parameter Eps and $MinPts$. In the dataset of *t8.8k*, if the region with smaller density (the second region with vertical shape from right in Figure 3.13) is successfully classified, there will be misclassification in other regions and vice versa, a referred paper [54] also shows the same results.

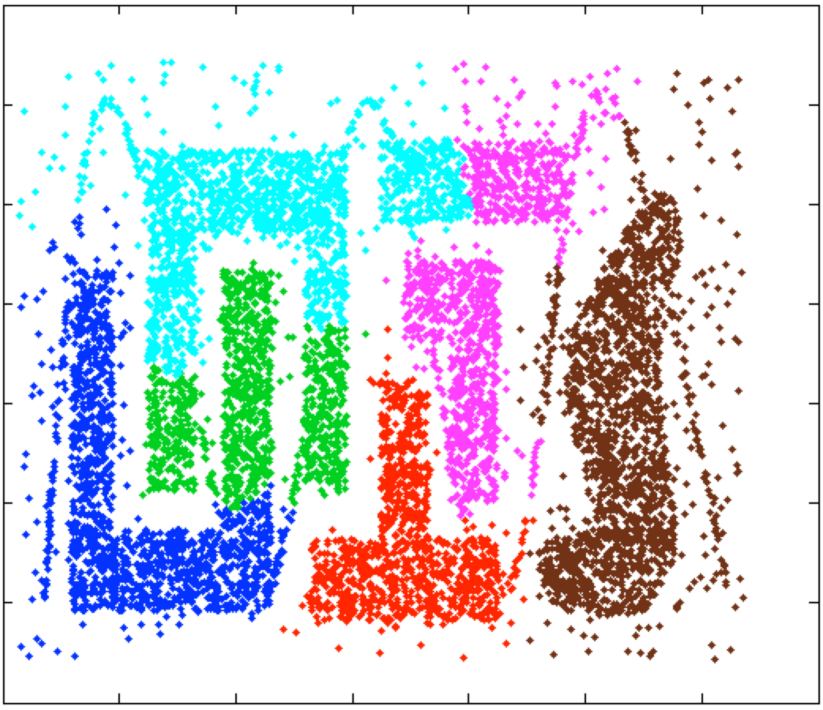

Figure 3.9 The clustering result of chmeleon t4.8k using FSFDP.

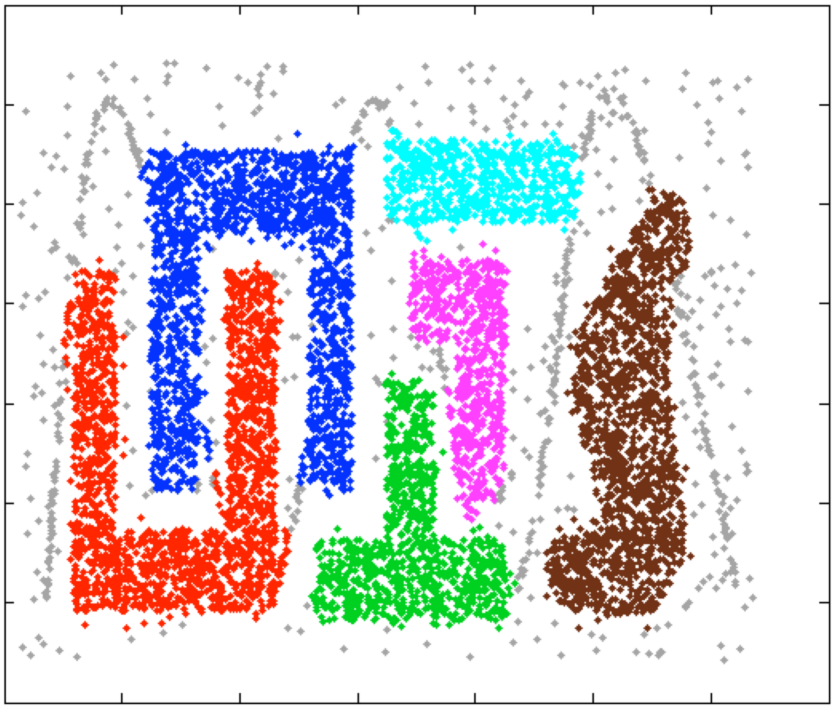

Figure 3.10 The clustering result of chameleon t4.8k using DBSCAN.

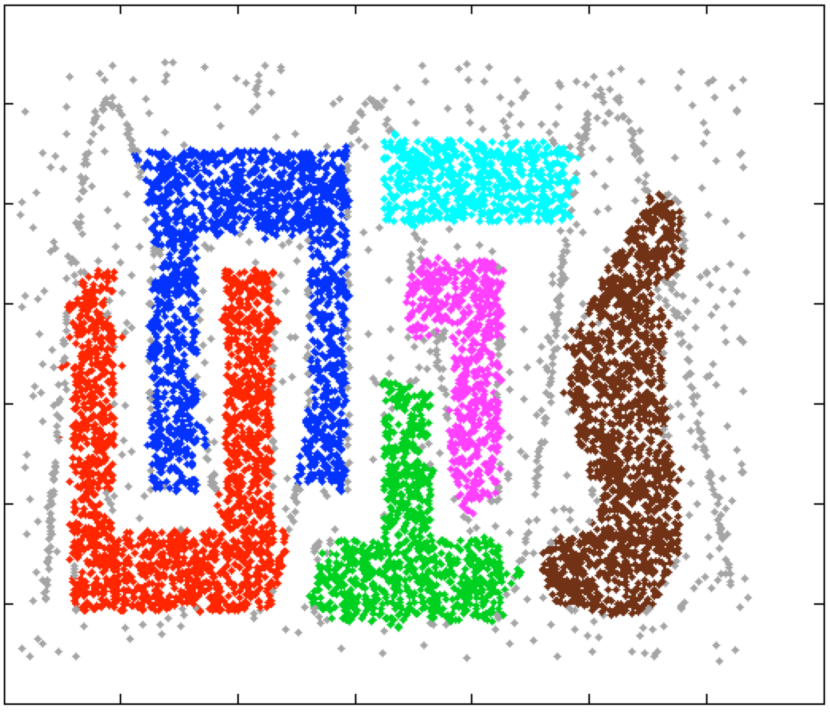

Figure 3.11 The clustering result of chameleon t4.8k using the presented DGB clustering method. Gray dots are the noise. Note that the white noise and even the sin-shaped noise are detected.

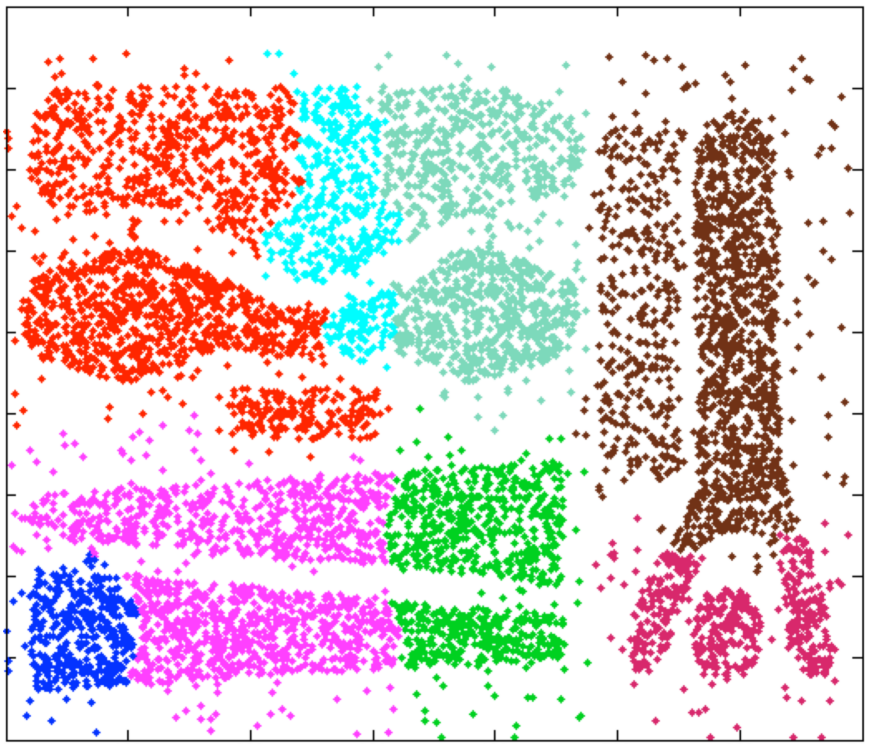

Figure 3.12 The clustering result of chameleon t8.8k using FSFDP

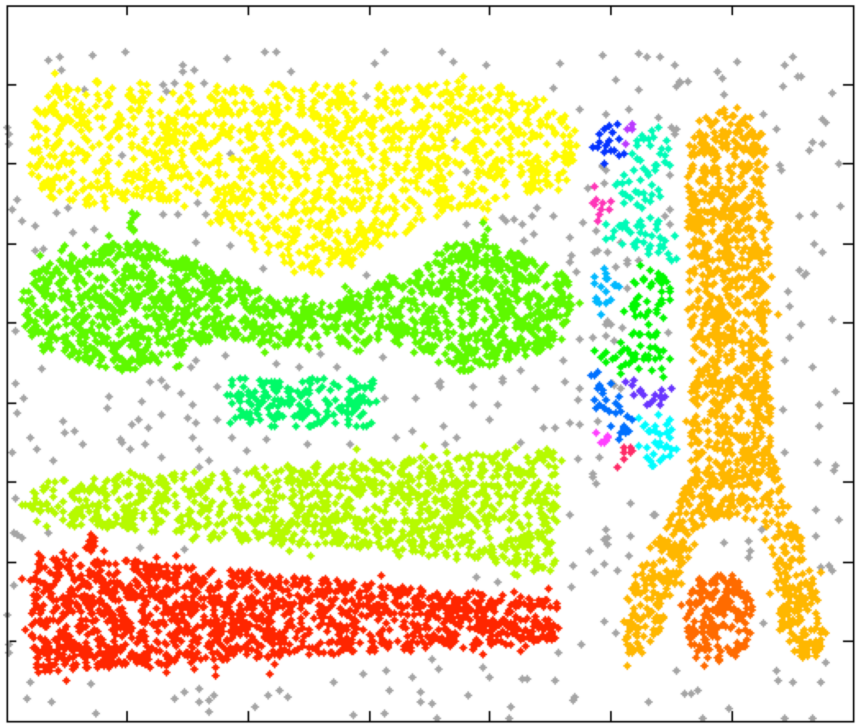

Figure 3.13 The clustering result of chameleon t8.8k using DBSCAN

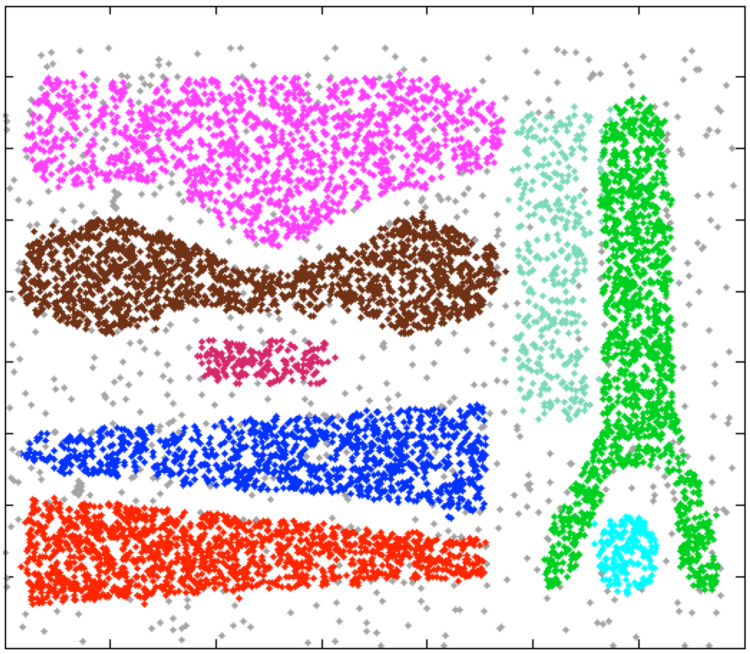

Figure 3.14 The clustering result of chameleon t8.8k using the presented DGB clustering method. It shows that the presented method is capable of classifying the dataset with different densities of clusters in it. Gray dots are the noise. Note that the white noise is detected.

3.4.3 Processing Time Analysis

For a given number of patterns, the time complex of the original DBSCAN is the same to FSFDP, which is $O(n^2)$, so the consuming time for these experimental datasets using the original DBSCAN is not listed in Table 3-1 but, in stead, an accelerated DBSCAN based on R*-tree search is applied. Table 3-1 lists the time comparisons among FSFDP, accelerated DBSCAN and the presented DGB clustering method, which shows that the presented method is much more efficient and effective than FSFDP and DBSCAN, and even than accelerated DBSCAN in some cases.

About the determination of grid size, one can not expect a universal grid size for every dataset. Neither too small nor too large grid size is selectable. Generally, the grid size is preferable but not limited if the node's local density figure (e.g. Figure 3.5) can illustrate the

outlines of mountain ridges. The computation time $O(n * nznode)$ ($nznode$ is the number of non-zero grid nodes) in processing these four experimental datasets using the proposed DGB method is shown in Figure 3.15 with the increase of grid size. In most cases, the processing time of different grid sizes will not change dramatically. For the dataset of *t8.8k*, the consuming time goes up because of the increase of non-zero-density nodes along with the increase of grid size.

Table 3-1 Processing time comparisons of four datasets.

Dataset	FSFDP	DBSCAN*	Presented DGB Method
FLAME (np=240)	0.2107 sec	0.0159 sec	0.0109 sec (11 *11 grid)
3spiral (np=312)	0.3527 sec	0.1654 sec	0. 2057sec (22 *22 grid)
t4.8k (np=8,000)	354.4838 sec	2.5660 sec	0.8941 sec (40 *40 grid)
t8.8k (np=8,000)	360.2049 sec	2.7024 sec	18.9509 sec (70 *70 grid)

(DBSCAN*: accelerated DBSCAN based on R*-tree search.)

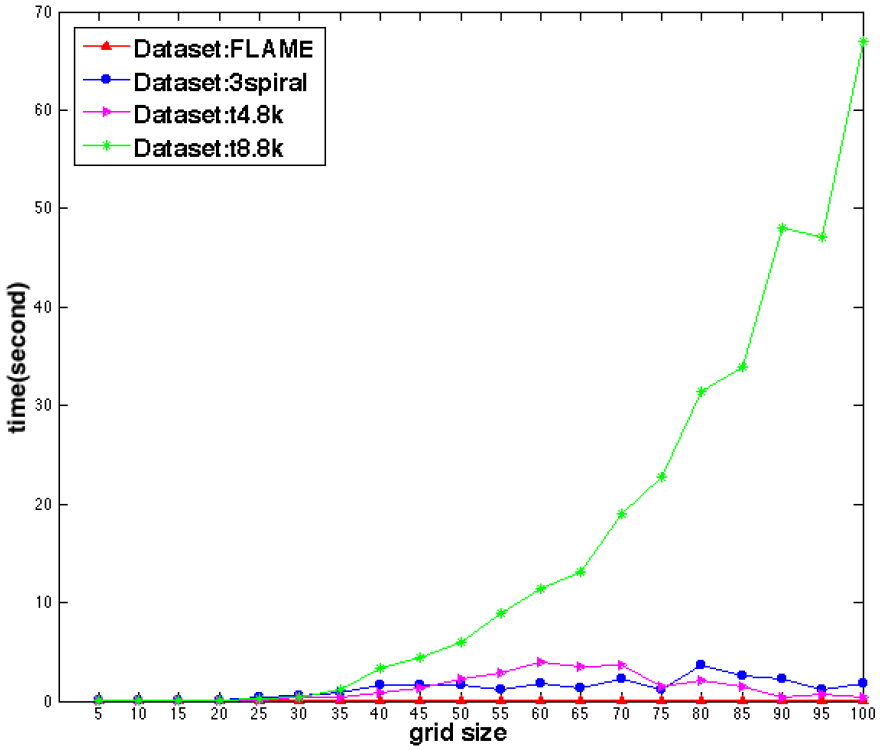

Figure 3.15 Computation time (in seconds) comparisons of four experimental datasets with different grid sizes.

3.5 Conclusions

Introduced in this chapter is a method called density and grid based (DGB) clustering method for clusters with arbitrary shapes, which is fast and feasible to classify data into different categories. Instead of giving a specific desired number of clusters, the presented DGB method finds clusters automatically. By setting a density threshold as noise, the proposed method is also capable of detecting white and non-white noise (the gray dots in Figure 3.7, Figure 3.11, and Figure 3.14). Without the calculation of Euclidean mutual distances between patterns, it successfully decreases enormously computation complexity to equation $O(n * nznode)$. A soft decision strategy using fuzzy approximation is proposed to compute node's density instead of simply counting the number of patterns in the cell or computing all other patterns' density contributions. Besides, a novel strategy of finding mountain ridges as the outlook of a cluster is explained. Moreover, specific classification of border patterns and noises is discussed as well.

It turns out that the presented DGB clustering method is capable of clustering datasets with arbitrary shapes, while the conventional K-means method (Figure 3.4) and FSFDP (Figure 3.9 and Figure 3.12) are not capable enough to do this job. Besides, there will be some misclassifications when the two parameters (Eps and $MinPts$) are contradictory to each other due to, e.g. the non-uniform density distribution (Figure 3.13) using DBSCAN. The experimental results (Table 3-1) of real-world datasets show that the proposed DGB clustering method significantly outperforms DBSCAN in the processing time due to the unnecessary computations of the Euclidean distances between mutual patterns, and the use of the standard grid and sparse matrix technique.

CHAPTER 4 CLUSTERING BY ANALYZING DENSITY CONSISTENCY AND MINIMUM INTERNAL AND EXTERNAL DISTANCE RATIO

Clustering is one of the most important tasks in preprocessing data, which is not easy to accomplish in just one single run especially for data with complex shapes and various dimensions. In this chapter, we will focus on a new clustering algorithm for clustering data with complex shapes based on density and the minimum internal and external distance ratio (DDR). With the concept of density and newly introduced cross-distance-ratio, the presented DDR classification algorithm developed a new two-stage strategy for accomplishing classification tasks. Besides, a more reliable analysis on the separation inside a large cluster and between clusters is described as well. The presented algorithm was successfully verified by classifying several datasets with various shapes and dimensions, i.e. human face photos. It turned out that the proposed DDR clustering algorithm is effective in clustering datasets into different categories with fewer misclassifications.

4.1 Introduction

In the previous chapters, we have mentioned various clustering approaches with pros and cons. For instance, K-means clustering method [1] is easily implemented; GMM-EM clustering algorithm [16] finds solutions iteratively; DBSCAN [13] and FSFDP [14] are two widely used density-based clustering techniques.

4.1.1 Partitional Clustering

Partitional clustering [55] decomposes a data set into a set of some sort of disjoint clusters. Given a data set of n points, a partitioning method constructs M ($M \leq n$) partitions of the data, with each partition representing a cluster. Specifically, it classifies the data into M

groups satisfying the following requirements: (1) each group contains at least one point, and (2) each point belongs to exactly one group. Notice that for fuzzy partitioning, which is not concerned in this chapter, a point can belong to more than one group.

Many partitional clustering algorithms follow the same strategies as the classification methods do mentioned in the section of Introduction. For example, in K-means and K-medoids partitional clustering methods, it tries to minimize an objective function that computes the sum of distances from data to the centers.

4.1.2 Density Based Clustering for Data with Various Shapes and Dimensions

Density-based classification approaches are favorable in clustering datasets with complex shapes and various dimensions, i.e. [13][14]. These approaches proceed classification in a transformed density domain while some other methods, e.g. K-means etc., operate in geometry domain. In density-based classification methods, datasets are classified to one category if they have similar local densities within a region. For each data, its local density can be computed using kernel functions, for instance, the radial basis function kernel, which is given as

$$\rho(i) = \sum_{j=1}^{n} e^{-\frac{(D_{ij}+C)^2}{2\sigma^2}} \tag{42}$$

where $\rho(i)$ is the local density or density, for simple, of data point i, n is the number of data points, D_{ij} is the distance between data point i and j, C and σ are constants.

4.2 Clustering by Analyzing Density Consistency and Minimum Internal and External Distance Ratio

This chapter presents a new algorithm, called DDR clustering algorithm, to do classification based on density and a newly defined concept of ratio between the minimum internal and external distance so that can be used to determine to merge partitioned clusters or

not. A novel strategy for computing density and partitional clustering were developed. Based on that, this new classification algorithm is capable of categorizing data points with complex shapes with fewer misclassifications. It can also achieve inspiring results of classifying high-dimensional data, which makes it practical and effective to be used in real life. The main contributions of this chapter are summarized as follows:

1) We introduced minimum internal and external distance ratio to the presented algorithm.

2) A density-based partitioning clustering is described.

3) In order to determine merge partitioned clusters or not, we utilized the curves of minimum internal and external distance ratios, as well as data density.

4.2.1 Minimum Internal and External Distance Ratio

In density-based classification approaches, datasets are categorized based on their local densities. Specifically, in the resulted clusters, data will have similar densities in a small region in a same cluster. And, between different clusters, the densities of datasets will be discrepant. What if in the raw classification results, the densities of data in adjacent clusters are similar to each other? Should they be merged to a large cluster or not? Let's denote the minimum internal and external distance ratio or cross-distance-ratio, for short, as the ratio between the minimum internal and external distance at first.

$$R_d = \frac{\min\left(D_{int}\right)}{\min\left(D_{ext}\right)} \tag{43}$$

where R_d is the cross-distance-ratio, D_{int} is the internal distance among the same cluster, and D_{ext} is the external distance to another different cluster.

One may notice that for each data point in current cluster, there will be a calculated R_d between current cluster and another cluster. Therefore, the number of $\{R_d\}$ in one cluster is equal

to the number of data points in the cluster. Moreover, for a given data point P, the min (D_{int}) from P to other data in the same cluster is unchanged when computing the R_ds. However, the min (D_{ext}) will depend on which cluster we consider. For example, in Figure 4.1, the closest data from Cluster II to data P is Q while it will be R from Cluster III. Besides, the min (D_{ext}) is not convertible. For instance, the closest data from Cluster II to data P, which is from Cluster I, is Q, while the closest data from Cluster I to Q is not P but S.

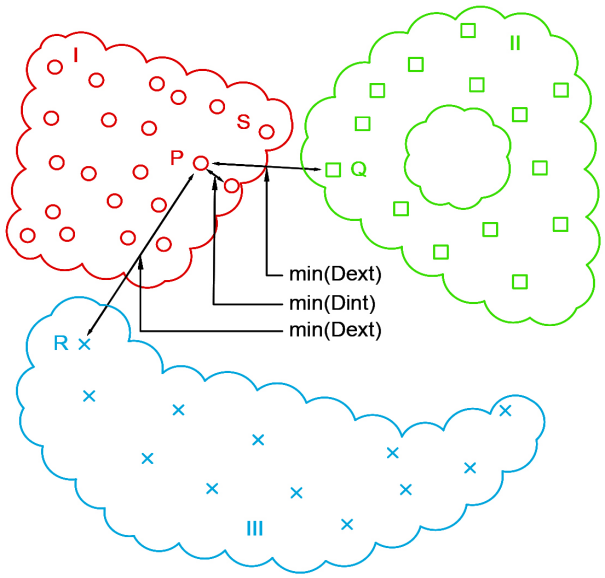

Figure 4.1 An instance of min (D_{int}), min (D_{ext}) in cross-distance-ratio.

4.2.2 Density Based Partitional Clustering (DPC)

In the first of two stages of our proposed method, a new density-based clustering is introduced. Datasets are categorized based on their densities and relationships in density-based classification methods. When two data points are close to each other, which may be measured in i.e. Euclidean distance, Manhattan distance, cosine similarity etc., they contribute more density to each other than those that are far away from each other.

The following density-based classification approach is served as partitional clustering. With the assumption that nearby data points belong to a same cluster if their local densities are

similar, this chapter presents a new strategy in four steps to do partitional clustering on datasets with various shapes and dimensions.

Step 1: Calculating mutual distances between data points. Distances are saved in distance matrix for fast processing in the following steps.

Step 2: Calculating data point's local density ρ_i as follows.

$$\rho_i = \sum_{j=1}^{N} \frac{1+1/D(i,j)}{1+D(i,j)^2} \tag{44}$$

where, N is the number of data, $D(i,j)$ is the distance between data point i and j. Different from the density computation using radial basis function kernel in (42), the contribution to data point's local density from its near data will be more significant using (44), which can provide us more accurate density calculations with an illustration in e.g. Figure 4.13. Other neural networks based density estimation techniques [56] can be explored as well.

Step 3: Finding out the neighbor information of data point i in density field. The minimum distance δ_i to data point i from a data point (i.e. data point j) with a higher local density will be found. We'll say that data point i is **pointed** to data point j or that data point j is **connected** with data point i in density field. Note that data with lower local densities are always pointed to data with larger local densities. For the point with the largest local density, its δ will also be set as the max. This step can be accomplished very fast utilizing the distance and local density information generated in Step 1 and Step 2.

Step 4: Clustering data, which will generate partitioned clusters. Starting with unclassified data point with the largest local density, which is defined as *density center*, if data point(s) **pointed** to it is within its n_c (4) closest neighbors, then data point(s) will be pushed into the same queue which initially contains the density center and mark the density center classified.

Conduct this processing until all the data points in the queue have no other data points pointed to within their n_c closest neighbors and mark them classified. All the data in the queue form a new partitioned cluster.

All the partitioned clusters will be obtained after all data are processed and classified according to Step 4.

$$n_c = round(N * perc) \tag{45}$$

where $perc$ is the percentage of closest neighbors, which is set to be as small as 5% or so in general case so that no misclassification will occur and only merging partitioned clusters is needed if any. The pseudocode for DPC algorithm is shown in Figure 4.2.

```
Input: raw data
Output: partitioned clusters pc
Algorithm:
run density-based partitional clustering:
 computing data mutual distances;
 computing data local densities (3);
 finding out neighbor information;
 initializing raw data as unprocessed;
 pc=[]: store partitioned clusters in rows;
 npc=1: initialize number of partitioned clusters;
 while exist unprocessed data do
    c=density center among the existed unprocessed
     data;
    mark c as processed;
    ccq=[]:initialize current cluster queue;
    while ccq is NOT empty do
       foreach data i ∈ raw data do
          if i unprocessed & i pointed to ccq[1] & i is
           within ccq[1]'s n_c (4) closest neighbors then
             ccq=[ccq i];
             mark data i as processed;
          end
       end
       ccq[1]=[]:pop out ccq[1];
       pc[npc, end + 1]=ccq[1]: store ccq[1] to current
        pc;
    end
    npc=npc+1: add another partitioned cluster;
end
```

Figure 4.2 First stage of the proposed DDR classification method: DPC algorithm.

After DPC process, data will be categorized initially, which brings satisfying results (e.g. Figure 4.4) and even perfect results (e.g. Figure 4.3) in some cases which need no more further processing.

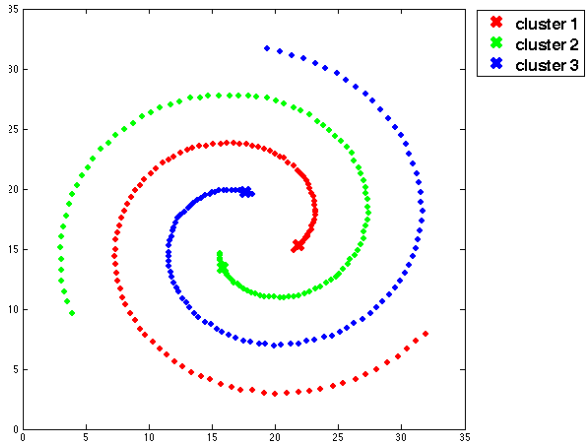

Figure 4.3 Clustering result using the proposed DPC method for *3spiral* dataset [52]. The crosses (x) are the starting data points for finding each partitioned cluster, which are the density centers in each cluster.

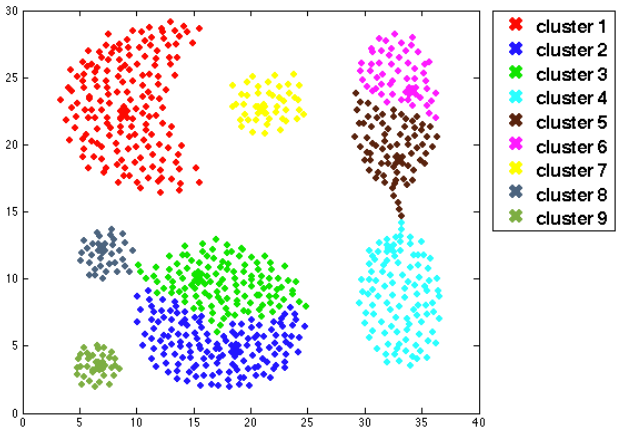

Figure 4.4 Clustering result using the proposed DPC method for Aggregation dataset [41]. The crosses (x) are the starting data points for finding each partitioned cluster, which are the density centers in each cluster. It shows that two large clusters are classified into two smaller clusters for each case and no misclassification happens. It will be further processed in the second stage of merging partitioned clusters to get final results.

4.2.3 Merging Partitioned Clusters Based on the Ratios of Minimum Internal and External Distance

In this second stage of the presented DDR classification method, merging the partitioned clusters generated in the first stage is executed if needed. The necessary of this stage depends on the ratios of minimum internal and external distance (Section 4.2.1).

Why cross-distance-ratio R_d is used as a criterion of merging near partitioned clusters? One may notice that the cross-distance-ratio gives some sort of neighbor information between partitioned clusters. Suppose all cluster information are given. Then, inside a cluster, data have similar mutual distances within a small radius. While between clusters, data will have much larger mutual distances than intra-cluster mutual distances. In other words, if two or even more partitioned clusters belong to a same larger cluster, then there will be R_d values between them close to 1 (>0.7, for instance), which are generated from the adjacent data between these partitioned clusters. On the other hand, if two or even more partitioned clusters DON'T belong to a same larger cluster, then all the R_d values between them should be much smaller than 1 (<0.6, for instance). Generally, the larger the R_ds are, the closer two partitioned clusters will be. Therefore, by examining the R_d values between two partitioned clusters, we can make a decision on merging them into a larger cluster or not. Two instances on the cross-distance-ratios between the partitioned clusters in Figure 4.4 are shown in Figure 4.5 and Figure 4.6.

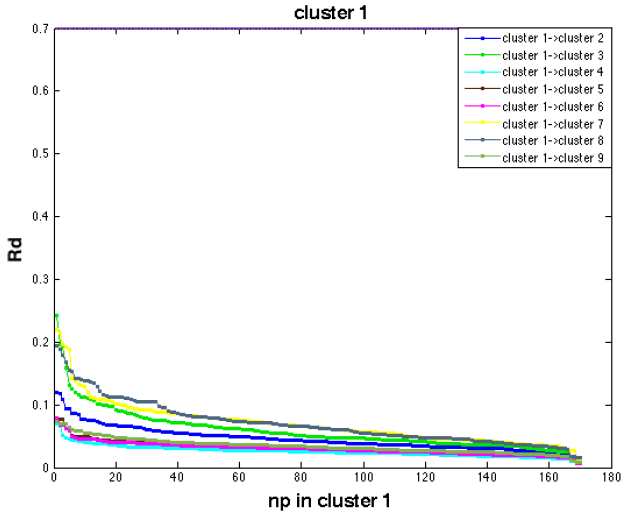

Figure 4.5 The cross-distance-ratios (R_d) between the 1st partitioned cluster and other partitioned clusters shown in Figure 4.4. R_d=0.7 is set as a reference line. It shows that all the R_ds are small, which means that the 1st partitioned cluster is separated away from other partitioned clusters and itself should be a cluster without merging other partitioned clusters.

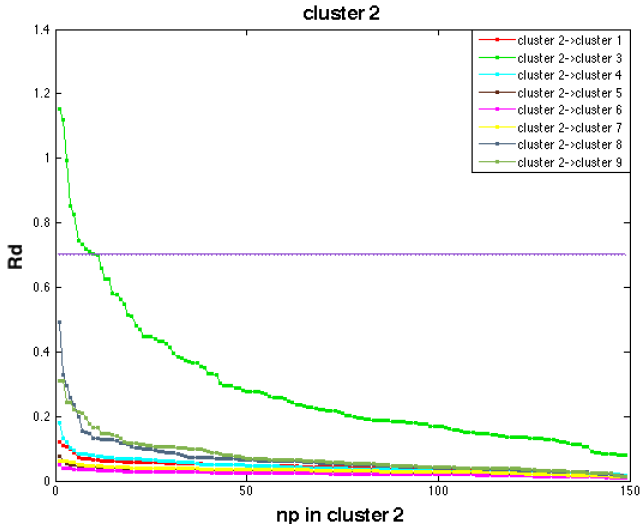

Figure 4.6 The cross-distance-ratios (R_d) between the 2nd partitioned cluster and other partitioned clusters shown in Figure 4.4. R_d=0.7 is set as a reference line. It shows that all the R_ds between the 2nd and 3rd cluster are significantly large with the largest R_ds around 1, which means that the 2nd partitioned cluster should be merged with the 3rd cluster. And, other R_ds are small, which means that the 2nd partitioned cluster is separated away from other clusters.

4.2.4 Identification of Separation Inside a Large Cluster and Between Clusters

For the case of a large cluster divided as several smaller partitioned clusters, i.e. cluster 2 and 3 in Figure 4.4, the cross-distance-ratios R_ds between them will be significantly different and large, shown in Figure 4.6. However, in some cases it may result with some amount of large R_ds but not significant for all of them (Figure 4.7), which are not supposed to merge those partitioned clusters. Therefore, for those large R_ds although not significant for all of them, it's necessary to identify whether they're representing a separation inside a large cluster or, in most cases, between clusters.

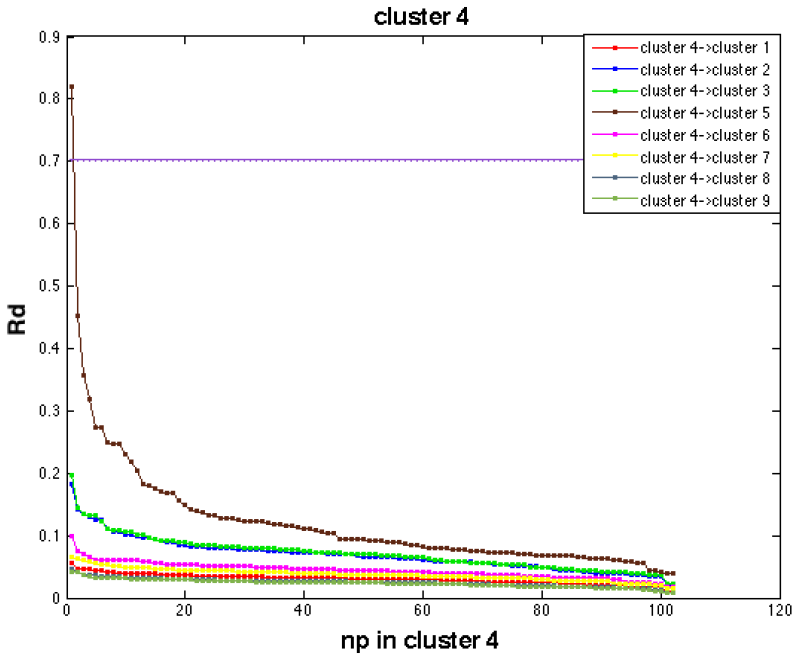

Figure 4.7 The cross-distance-ratios (R_d) between the 4th partitioned cluster and other partitioned clusters shown in Figure 4.4. R_d=0.7 is set as a reference line. It shows that there is one large R_d value between the 4th and 5th cluster, while the rest of R_ds remain small. This large R_d is a reflection of border conncetion between the 4th and 5th cluster, which should not belong to one large cluster because of the large difference between internal density and external density.

Suppose there are short connections/distances between two partitioned clusters at border data points. So at the border data points, the R_d values will be large, which are caused by the short distances between these two partitioned clusters. However, if these two partitioned clusters should not be merged into a larger cluster, the local densities at the border data points from one partitioned cluster are significantly different with the local density inside the other partitioned cluster. Therefore, it can be used to identify whether the large R_ds are reflecting a separation inside a large cluster or between clusters based on their densities. As we have known the *density centers* of each partitioned cluster, data local densities as well as the data mutual distances at the first stage of DPC (Section 4.2.2), it's easy to compute the average local density of data, which have distances to border point no larger than the distance from border data point to the *density center* of the same cluster. This average local density is called *the external density* of border point when it's treated as the external partitioned cluster in calculating the cross-distance-ratios R_ds. Notice that *the internal density* of data are exactly the local density computed in the first stage of DPC.

So, for those data points that generate large R_ds, by examining the difference between one data point's *internal density* and the other data point's *external density*, we can determine whether these two data points are adjacent data inside a larger cluster or the border data of two different clusters. Specifically, the *internal density* of one data point is similar to the *external density* of the other data point, i.e. $1 \pm 20\%$ of the involved density, when they are adjacent data points between two partitioned clusters, which are actually should be merged into a larger cluster. For the border data points, the *internal density* of one data point is significantly different to the *external density* of the other data point, i.e. $1 \pm 50\%$ of the involved density.

For the partitioned cluster that has only one data point, there is no internal distance. In that case, the internal distance is set to be its minimum external distance such that its max R_d is equal to 1. Then, as usual, calculate the cross-distance-ratio R_ds and compare its internal density with its near data points' external density to determine the merging operation or isolating itself as an outlier. Note that the compared near data points are the data that produce large R_d values (i.e. $R_d > 0.7$).

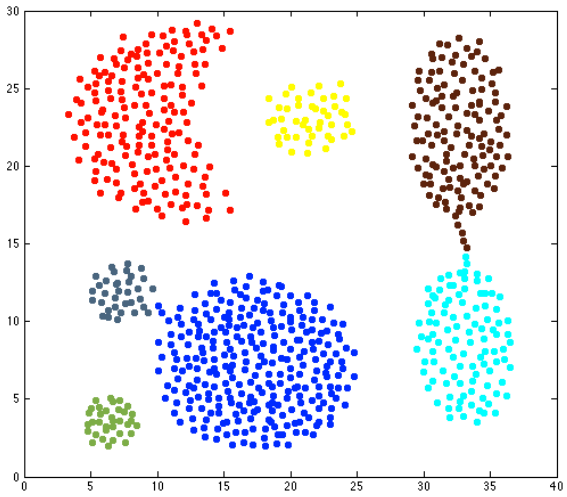

Figure 4.8 The final result of Aggregation dataset using the presented DDR clustering method. All clusters are correctly presented.

4.2.5 Time Complexity Analysis

For the best case of accomplishing the task in the 1st stage of our method, the time complexity is $O(N^2)$ due to the computation of mutual distances. Now, let's discuss the worst case where each data point is isolated as a partitioned cluster after 1st stage of our method. In this case, we need to check the density consistency between every two partitioned clusters in the 2nd stage, which has time complexity of $O(N(N-1))$. So the total time complexity is $O(N^2) + O(N^2 - N)$. Generally, the complexity of the proposed clustering algorithm is

$$O(N^2) \sim O(N^2) + O(N^2 - N) \tag{46}$$

4.3 Experimental Results

Three datasets with various dimensions and numbers of clusters have been used to test the presented algorithm in this section. All the experiments are implemented based on the same software and hardware: Matlab R2013a in OS X operating system with Intel Core i5 (I5-4258U) @2.4GHz 8.00GB memory.

4.3.1 FLAME

The DNA microarray data of *FLAME* [39] is tested for our method. With the accurate density calculations of each data point, the presented DDR classification algorithm is capable of identifying the correct boundary between the two main clusters with different shapes, shown in Figure 4.9.

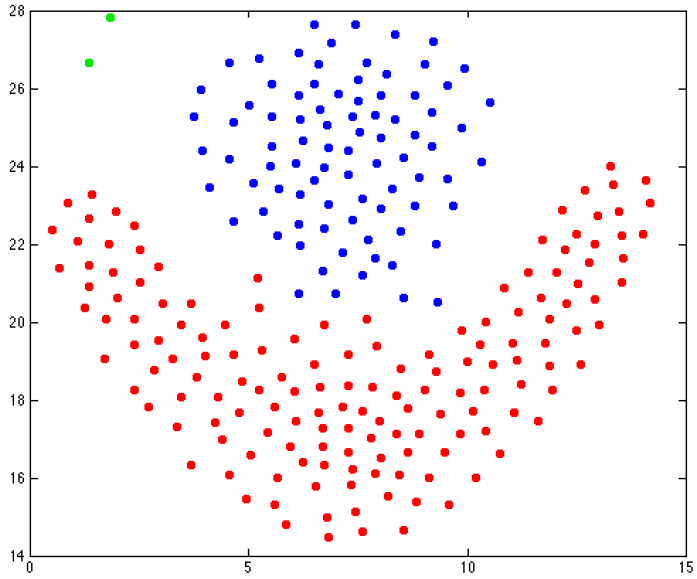

Figure 4.9 Classification result of FLAME dataset using the presented DDR classification method. The two large clusters are correctly presented as well as the upper-left outliers.

4.3.2 2Spiral

Classifying the complex-shape *2Spiral* is a challenging job for a classification method. It is much harder than classifying *3Spiral* (Figure 4.3) data because the distances between clusters

at the outside region are even smaller than the mutual distances among the same cluster. Therefore, the distance-based classification approaches may fail to accomplish this job. Different parameter settings will generate different *2Spiral* datasets, e.g. [57]. The specific *2Spiral* data used in this chapter is generated according to the expressions below in Figure 4.10 with 97 data points for each spiral.

```
c = 1.3; //the larger, the harder to classify 2spiral
for i=1: 96
        angle = i*c*3.1415926/16;
        radius = 6.5*(104-i)/104;
        xA = radius*sin(angle);
        yA = radius*cos(angle);
        xB = -radius*sin(angle);
        yB = -radius*cos(angle);
end
```

Figure 4.10 The psuedo code of generating 2Spiral data that is used in this section.

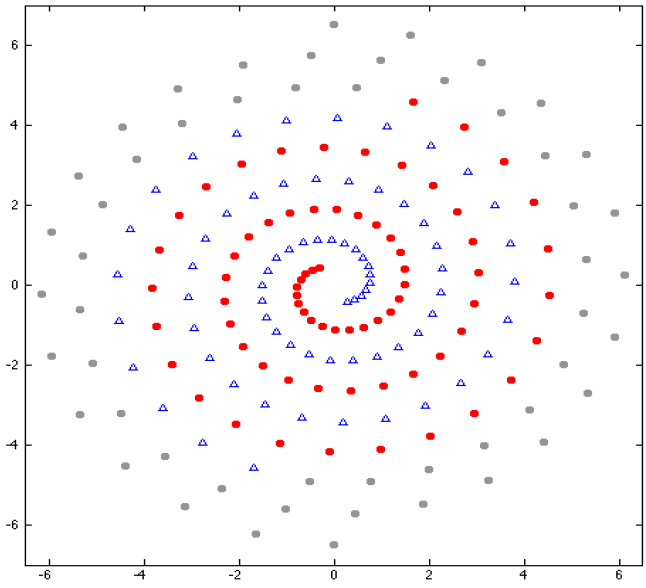

Figure 4.11 The classification result of 2Spiral dataset using DBSCAN. The required best parameter setting are: $Eps = 1.24$, $MinPts = 2$. As it shows, DBSCAN left many outside data points unclassified or treated as noise, which are marked with grey color dots.

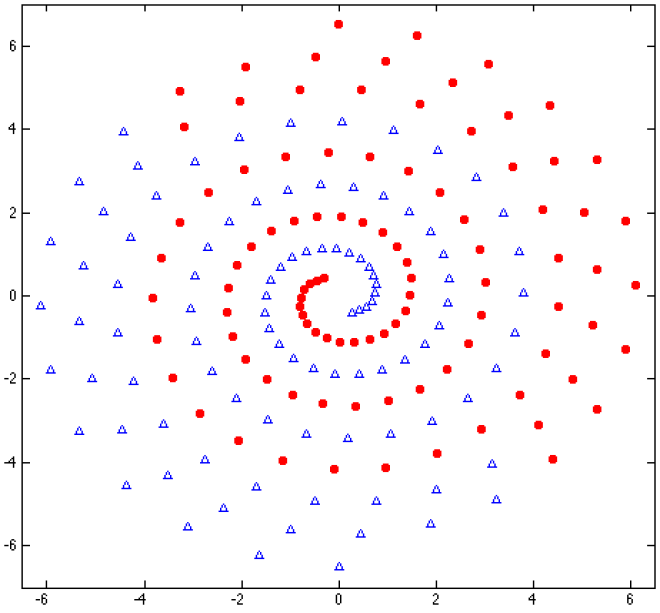

Figure 4.12 The classification result of 2Spiral dataset using FSFDP. It classified the outside data points but with wrong labels.

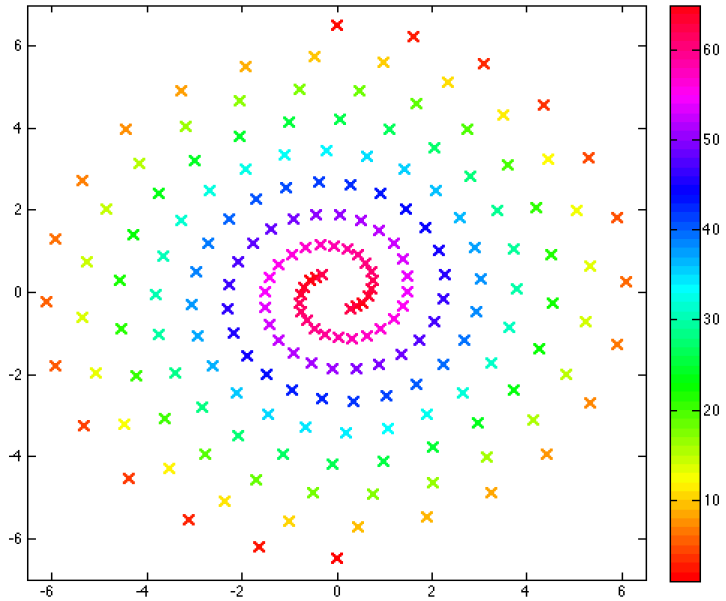

Figure 4.13 Classified 2Spiral data using the proposed first-stage DPC method. It shows that no misclassification occurs as each data point is classified as a partitioned cluster, marked as X. Color information of each cluster indicates its density scale, which is found smoothly changing between should-be neighbors. In the following stage of merging partitioned cluster if needed, 2Spiral will be clustered eventually using the proposed second-stage method.

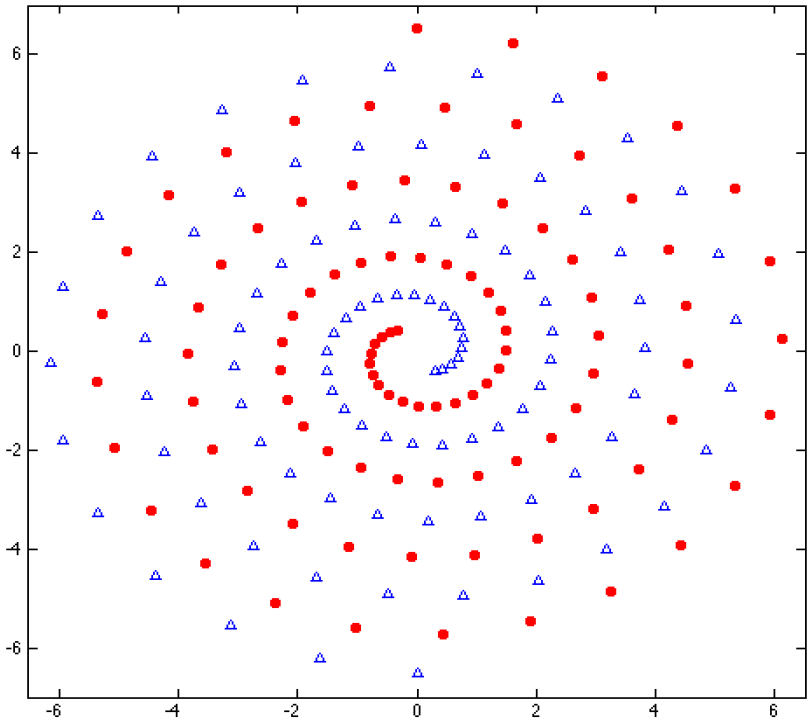

Figure 4.14 The final classification result of 2Spiral dataset using the presented DDR clustering method. Notice that the centroid-based methods, e.g. K-means classification method, are not capable of classifying this dataset.

4.3.3 Olivetti Face

Face datasets classification is explored in many literatures [58] [59]. Olivetti Face data [60] is a set of 112*92-pixels (or 10,304 dimensions) scaled images of different persons with different face angels, facial expressions (open/closed eyes, smiling/no-smiling) and even with or without glasses wearing. It has complex shapes as well although one can not draw them in 10,304 dimensional space. Unlike the time costly convolutional neural networks [6] utilizing the image content features, our proposed classification method is capable of providing us inspiring results (Figure 4.16) with a much shorter processing time based on data distances. In Figure 4.16, 10*10 Olivetti Face images are classified as a 10-10-10-10-10-10-10-10-10-9-1 cluster. Before that, those 100 Olivetti Face data was clustered as 5-5-8-1-1-10-5-2-2-1-5-5-6-3-1-10-5-5-10-7-

1-1-1 using the presented first-stage DPC method with no misclassification occurs and only merging some clusters is needed on them, shown in Figure 4.15.

Figure 4.15 The presented first-stage DPC clustering result of 100 Olivetti Face data. Face images in the first column are the density centers, which present moderate face features in each category. The red face images are partitioned clusters that need to be processed in the second stage. It shows a 71% classification correction using the presented first-stage DPC method. Note that the classification correction for these 100 images using FSFDP is 41% in [14].

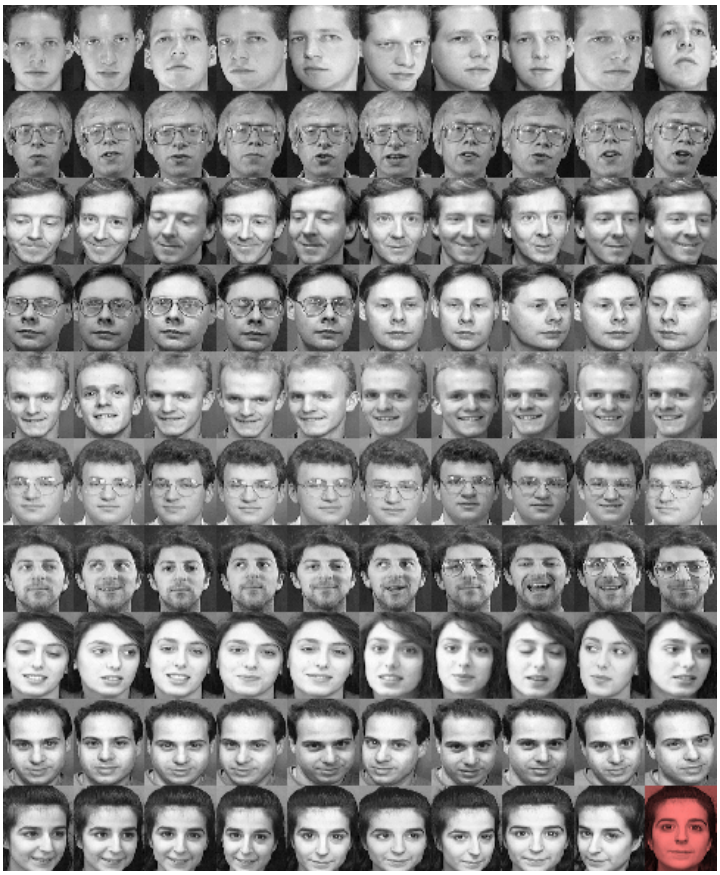

Figure 4.16 Classification result of 100 Olivetti Face data using the presented DDR method with a total processing time of 0.5635 seconds. It shows an impressive classification result and 99% faces are correctly classified. Only one face image marked with red color in the last row is classified as an outiler.

4.3.4 Evaluation Metric

The standard unsupervised evaluation metric and protocols for evaluations and comparisons to other algorithms are used. Intuitively, this metric considers a cluster assignment from an unsupervised algorithm and a ground truth assignment and then finds the best matching between them. The best mapping can be efficiently computed using the Hungarian algorithm [61]. For all the approaches, the number of clusters is set to be the number of ground-truth categories. Clustering performance is evaluated with *unsupervised clustering accuracy (ACC)*:

$$ACC = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n} \quad (47)$$

where l_i is the ground-truth label, c_i is the cluster assignment produced by the algorithm, and m ranges over all possible one-to-one mappings between clusters and labels.

4.3.5 Evaluation Results

Comparisons on classification results on mentioned datasets using involved classification algorithms are shown in Table 4-1. Instead of presenting the final results as one, results of the two stages in the proposed DDR classification algorithm are illustrated respectively in Table 4-1. From Table 4-1, we can see that spectral, FSFDP and DBSCAN clustering methods works well with datasets with non-complex shapes. However, when the shape of datasets get complex, they are not efficient to give us satisfying results.

Table 4-1 Comparisons of clustering accuracy (ACC) and processing time on seven datasets.

Method		*3Spiral**	*Aggregation*	*FLAME**	*Jain [62]*	*Lsun [63]*	*2Spiral*	*Olivetti Face* *(N=100)*
K-means		31.73% (0.5035 sec)	86.93% (0.5064 sec)	83.75% (0.4955 sec)	78.02% (0.5000 sec)	76% (0.5075 sec)	50% (0.4833 sec)	84% (1.2887 sec)
GMM-EM		33.33% (0.0932 sec)	92.51% (0.1889 sec)	72.92% (0.1248 sec)	57.64% (0.0904 sec)	100% (0.1046 sec)	50.52% (0.0733 sec)	47% (0.6141 sec)
Hierarchical		35.58% (0.6316 sec)	99.62% (5.9409 sec)	83.33% (0.3618 sec)	94.64% (0.9126 sec)	72% (1.0002 sec)	53.61% (0.2538 sec)	32% (64.1721 sec)
Spectral		100% (0.0378 sec)	91.24% (0.0834 sec)	98.75% (0.0351 sec)	100% (0.0461 sec)	100% (0.0469 sec)	50% (0.0499 sec)	54% (0.7236 sec)
FSFDP		100% (0.0423 sec)	100% (0.1971 sec)	99% (0.0343 sec)	100% (0.0342 sec)	100% (0.0703 sec)	88% (0.0313 sec)	61% (0.0225 sec)
DBSCAN		100% (0.0133 sec)	99.75% (0.0510 sec)	97.08% (0.0068 sec)	92.76% (0.0178 sec)	100% (0.0152 sec)	73% (0.0105 sec)	78 % (0.1594 sec)
DDR	*1st-S*	**100%** (0.0384 sec)	78.6% (0.1571 sec)	**100%** (0.0295 sec)	37.80% (0.0367 sec)	64.50% (0.0514 sec)	2% (0.0386 sec)	71% (0.0420 sec)
	2nd-S	**100%**	**100%** (0.2577 sec)	**100%**	**100%** (0.5446 sec)	**100%** (0.5503 sec)	**100%** (0.4359 sec)	**99%** (0.5215 sec)

As a comparison, one has to select the centers in the decision graph when applying FSFDP. It can not present the results automatically without manual selection on centers, otherwise there will be even larger classification errors. Secondly, it's very difficult to find the required two parameters precisely, *MinPts* and *Eps*, when using DBSCAN because one may have no idea at all on how large the *MinPts* should be within what distance (*Eps*) in the beginning. You have to

do a lot of trial-and-errors on *MinPts* and *Eps* before finalizing them. Even though it costs the least time per run in Table 1 in most cases, adding the manual operation time to find the best parameters, the time is much longer using DBSCAN than our method. Besides, it pays more attentions on data's connections instead of data's densities in DBSCAN, which may cause larger classification errors (e.g. Figure 4.11). Although the total processing time of the presented DDR classification algorithm is not prominent, the first stage of it is competitive to others. All in all, it is not an easy job to seek a higher classification correction as well as guarantee a shorter processing time. With the employment of new concept of cross-distance-ratios, the presented DDR classification method is capable of providing us impressive classification results as shown in Table 4-1.

4.4 Conclusion

This chapter presented a classification algorithm, called DDR classification method, for classifying data with complex shapes and various dimensions based on Density and the minimum internal and external Distance Ratio (DDR). As it is not an easy job to classify data in just a one single run, a novel two-stage classification strategy is developed in the proposed DDR method at the cost of slower speed. Using the new proposed first-stage density-based partitional clustering (DPC) method, it can provide reasonable/satisfying partitional clustering results (e.g. Figure 4.4) and even perfect results as desired (e.g. Figure 4.3). In the second stage of the presented DDR classification algorithm, it will determine whether to merge some partitioned clusters or not based on the cross-distance-ratio curves. Besides, an effective way to identify the separation inside a larger cluster and between clusters is described as well by analyzing the external density and internal density.

Based on two facts that the densities are smoothly changing inside one cluster and the distances as well as the densities are significantly different between clusters, the proposed DDR classification algorithm is capable of generating satisfying results. For instance, the densities (44) are calculated constantly changing among a same cluster while the densities are significantly different between these two clusters in Figure 4.13. A new concept of minimum internal and external distance ratio, called cross-distance-ratio, is described and used to determine merging two partitioned clusters or not (e.g. Figure 4.6). For those with not significant large cross-distance-ratio curve (e.g. Figure 4.8), the merging took place depending on the difference between their internal and external densities. A good example applying that strategy is shown in Section 4.3.2, which works quite well. Besides, the proposed DDR classification method is capable of classifying high-dimensional data. For the first 100 Olivetti Face photos, it provides us a 99% correction on classifying them using DDR classification algorithm (Figure 4.16). Therefor, the presented DDR classification algorithm can be considered as a replacement of traditional complex shape classification algorithms.

CHAPTER 5 DATA VISUALIZATION

As previous discussed, clustering is one of the most important unsupervised learning techniques. Various clustering strategies come out for different clustering problems. Beyond that, in this chapter, we will demonstrate that clustering approaches can be also applied into data visualization. To be specific, we proposed an unsupervised multi-dimensional scaling (MDS) method to visualize high-dimensional data in a low-dimensional space, i.e. 2 or 3 dimensional (2D/3D) space. Different from traditional MDS approaches where the only purpose is to embed high-dimensional data into a low-dimensional space, this study aims at embedding data into a low-dimensional space as well as clustering them into small clusters, thus enlarging the margins between categories and providing better visualization. Considering the density relationships inherent in data, this chapter proposes a new density-concentrated multi-dimensional scaling (DCMDS) algorithm to perform data visualization. One benefit of the proposed DCMDS algorithm is the ability to embed data more accurately than traditional MDS techniques by using second-order gradient optimization instead of first-order gradient only. A key advantage of the presented DCMDS algorithm is the capability to concentrate categorical data, which enlarges the margins between data. In the resulted embedding, data are compact in clusters. The results demonstrate that the proposed DCMDS algorithm outperforms conventional MDS methods regarding to Kruskal stress factors. It can be easily extended into embedding in any desired low-dimensional space.

5.1 Introduction

In recent years, all kinds of data have expanded with the emergence of the Big Data era, and accordingly, the demand of understanding these data increases. As visualization techniques

can provide intuitive and vivid knowledge of data, many visualization approaches [64][65] are developed to understand emerging data in a faster and more informative manner.

A normally used method to achieve an accurate visualization of high-dimensional data is learning a low-dimensional embedding of the high-dimensional data. Low dimensional representation of data should reveal corresponding relationships in higher dimensions. Specifically, data in close proximity represent similarity and data separated by long distances represent dissimilarity.

Conventional visualization methods derive from dealing with the problem of dimensionality reduction. Different methods are proposed in dimensionality reduction techniques, such as Principal Components Analysis (PCA) [25], Nonnegative Matrix Factorization (NMF) [26] etc. Matrix transformations are taken to obtain the principal components in a smaller matrix fulfilling dimensionality reduction. In general, matrix operations are easy to be realized and can provide results quickly. However, these dimensionality reduction approaches are not capable of preserving the dimensionality information except the principal components. One can expect a general dimensionality reduction result but at the expense of meticulous embedding.

As a means of visualizing data while preserving dimensionality information in the form of distances, Multidimensional Scaling (MDS) techniques [30] are developed. A set of MDS techniques can be found in literatures, such as Isomap [11], locally linear embedding (LLE) [12], Sammon mapping [10], LAMP [33] etc. In general, an MDS algorithm is proposed to place data iteratively in low dimensional space such that the distances between data are preserved as well as possible. The majority of those techniques make different attempts to simulate the short pairwise distances between data, which are considered to be dependable in high-dimensional space. For example, LLE considers preserving only the local, small distances at the expense of not

including remaining distances. Furthermore, there is much uncertainty as to what defines the "local" range. Even though the famous Sammon mapping method optimizes all the mutual distances (i.e. not just local small, local distances), it suffers many overlaps between categories and is not guaranteed to converge. LAMP is one of the MDS techniques based on landmarks. Different landmarks give different MDS results. In summary, those traditional MDS methods still have the following shortcomings:

1) The optimization on distance preservation only uses the first-order gradient method, which is easily trapped in local minima.

2) Narrow margins among clusters create overlapping in the mapped results.

3) Most existing methods only consider distances between data in embedding, whereas additional factors could provide desirable information.

In this chapter, we revisit MDS techniques and ask: *Can we use an unsupervised learning approach to conduct MDS purpose for visualization and clustering jointly?* In order to improve the stated shortcomings of traditional MDS methods, optimization based MDS with unsupervised clustering may provide both a promising and effective solution. Optimization methods using second-order gradient descent have the ability to produce more accurate results than first-order methods due to their stronger ability to escape from local minima. However, second-order gradients often bring heavy computation load. In unsupervised learning, clustering is one of the advanced techniques that can provide data category information. Regarding to what the users want to get from the MDS results, cluster automatic formation is thought to be one of the highest demanded expectations. Other expectations include larger margins between clusters, individual data relationships, etc. Therefore, in order to utilize the second-order gradient approach to fulfill MDS purpose with cluster automatic formation as well as possible, this chapter proposes a new

Density-Concentration Multi-Dimensional Scaling (DCMDS) algorithm. The key idea behind the proposed DCMDS algorithm is to use density based clustering and incorporate it with Levenberg–Marquardt (LM) optimization based MDS. This algorithm presents an alternative MDS approach using LM optimization and density concentration, yielding improved MDS performance. The main contributions of this chapter are summarized as follows:

1) We propose a new unsupervised algorithm for general MDS purpose based on the LM optimization method. As the LM method can automatically switch between first-order gradient and second-order gradient optimization methods, the MDS technique using LM optimization shows great improvement in mapping data because of its ability to escape from local minima.

2) To obtain a better visualization on data category information, density based clustering is integrated in the MDS process. In the mapping results, data move based on their mutual distances as well as their density relationships. Because of this, better cluster gathering and larger cluster margins are achieved during mapping without knowing any category knowledge.

3) Our proposed algorithm is evaluated and compared with other MDS approaches, including Sammon mapping, Isomap, LLE and LAMP. When evaluated on experiments involving mapping several real life data on a 2D plane, the DCMDS algorithm outperforms traditional MDS approaches. Moreover, it has the ability to provide mapping results in any desired dimensional space.

The rest of this chapter is organized as follows. Section 5.2 briefly reviews the related concepts, followed by our detailed DCMDS algorithm for data visualization. Section 5.3 demonstrates the experimental results on several high-dimensional datasets to evaluate the effectiveness of the proposed algorithm. Finally, Section 5.4 draws the conclusions.

5.2 Density-Concentrated MDS Algorithm

MDS is a technique used for embedding high-dimensional data into a low-dimensional space where the distances in the low dimension well represent the distances in the original, high-dimensional space. As data with short distances are similar to each other, MDS can also be applied to analyze the similarity, dissimilarity, or relationships between data. The general cost function of MDS on a 2D plane is defined in (48), where n is the data number. The general goal of MDS approaches is to minimize the cost function.

$$Err = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij})^2 \quad (48)$$

In general, there exists two types of MDS algorithms: metric and non metric.

1) In metric MDS approaches, the actual values of the dissimilarities are used. The distances between datapoints are then set to be as close as possible to the similarity or dissimilarity data. Sammon mapping and LAMP techniques are two of the standard metric MDS approaches.

2) In non-metric MDS approaches, the algorithms will try to preserve the order of the distances, and hence seek for a monotonic relationship between the distances in the embedded space and the similarities/dissimilarities. In non-metric MDS techniques, such as Isomap, LLE, etc., most of the choices for criterion become undefined when two datapoints are at the same location ("co-located") because of the same distances between data. When "co-located" happens, it will suspend non-metric MDS processing without an MDS solution.

5.2.1 Density Based Clustering

Given the assumption that cluster centers are surrounded by neighbors with lower local densities and are at a relatively large distance from any data with a higher local density, density

based clustering methods [13][14] are capable of fulfilling unsupervised clustering task with satisfying results. There are three steps according to density based clustering approaches [14].

Step 1: Calculate the data local density ρ_i. Regarding to density computation, a Gaussian kernel based density contribution is given as (49).

$$\rho_i = \sum_{i \neq j} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)\right)^2 \tag{49}$$

Here, ρ_i is the local density of data i. d_{ij} is the distance between data i and j. d_c is the cutoff distance.

Step 2: Calculate the minimum distance δ_i between data i and any other data with a higher local density. The calculation of δ_i is given in (7).

$$\delta_i = \begin{cases} \min(d_{ij}), & \rho_j > \rho_i \\ \max(d_{ij}), & \rho_j < \rho_i \end{cases} \tag{7}$$

Here, $j = 1,2,\dots,np \; and \; j \neq i$.

Step 3: Generate the decision graph. Aim to choose the data with large local density ρ_i as well as large minimum distance δ_i as the cluster centers. The decision graph is generated with x-coordinate is ρ_i and y-coordinate is δ_i.

After the selection of cluster centers in the decision graph, data will be assigned into different clusters based on the minimum distances δs.

5.2.2 Density-Concentrated MDS Algorithm for Data Visualization

The proposed algorithm focuses on projecting high-dimensional data into a low (2D/3D) space. The overall algorithm of the proposed DCMDS is illustrated in Algorithm 5-1. The benefits of the proposed DCMDS algorithm include applying first-order and second-order gradient descent methods to optimize data locations and using density relationships between data

to concentrate clusters and enlarge margins between them. It simultaneously learns MDS embedding and clustering.

A. MDS process in the proposed DCMDS algorithm

The cost function of MDS methods in (48) leads to an alternating non-linear least-squares optimization process, where we alternate between re-computing different data, and each step and iteration is guaranteed to lower the value of the cost function. In most cases, the optimization process is fulfilled using first-order instead of second-order gradient approaches, allowing the process to be trapped in local minima. The Levenberg-Marquardt (LM) method, which is developed to solve non-linear least squares problems iteratively, finds the best solutions by switching between first-order and second-order gradient approach via a damping parameter. Unlike the second-order gradient methods, which are of heavy computation, the LM method approximates the second-order gradient with the first-order gradient. In this chapter, the LM method is adopted as the MDS technique to determine data positions. In the proposed DCMDS algorithm, MDS embedding process has two phases: (1) data position initialization using matrix eigen-decomposition and (2) data postion optimization using LM method.

1) Getting initial data positions for LM method via matrix eigen-decomposition.

Instead of randomly generating initial positions, the matrix eigen-decomposition technique is used to provide us initial positions very fast on a 2D plane or 3D space. It can accelerate the non-linear least squares optimization process.

Suppose DMat is the distance matrix that contains all the between-data distances and $\mathrm{DMat}(\mathrm{i}, \mathrm{j})$ returns the distance between data i and j. Then, the initial data positions on a 2D plane are give as (56) via matrix eigen-decomposition.

$$SD = DMat.* DMat \tag{50}$$

$$ti = sum\big(SD(i,:)\big) \tag{51}$$

$$tj = sum\big(SD(:,j)\big) \tag{52}$$

$$ta = sum(sum(SD) \tag{53}$$

$$M(i,j) = \frac{1}{2}\big(ti + tj - ta - SD(i,j)\big) \tag{54}$$

$$[V, D] = eigs(M) \tag{55}$$

More detailed $eigs$ operation can be found in [66][67]. Diagonal matrix D contains the eigenvalues on the main diagonal. The columns of matrix V are the corresponding eigenvectors. Note that the first eigenvalue $D(1,1)$ is zero, which should be neglected.

$$P^{(0)} = \sqrt{D(2:3,2:3)} * V(:,2:3)' \tag{56}$$

Next, with the initial positions, embedded data is optimized iteratively using LM method.

2) LM method

LM method [68] is applied for minimizing the least-square cost function (48) in the DCMDS algorithm. In order to return better gradient-based optimization results, the second-order derivatives of the total error function are considered. However, the calculation of Hessian matrix H, which contains the second-order derivatives of cost function, is often complicated. In order to simplify the computing process [69], the Jacobian matrix J is introduced to approximate Hessian matrix H. J is the matrix of all first-order derivatives of the cost function with respect to data's coordinates. For the cost function, the m'th row of Jacobian matrix is $J_{m:} = [\frac{\partial Err}{\partial x_m} \frac{\partial Err}{\partial y_m}]$.

$$H \approx J^T J \tag{57}$$

In order to make sure that the approximated Hessian matrix $J^T J$ is invertible, LM algorithm introduces another approximation to Hessian matrix:

$$H \approx J^T J + \mu I \tag{58}$$

where μ is combination coefficient with positive value; I is the identity matrix with the size of 2*2 for 2D MDS. From equation (58), one may notice that the elements on the main diagonal of the approximated Hessian matrix will be larger than zero. Therefore, with this approximation, it can be sure that matrix H is always invertible.

Now, the update rule of Levenberg-Marquardt algorithm can be presented as follows.

$$\Delta = (J^T J + \mu I)^{-1} * J * Err \tag{59}$$

$$x = x + \Delta(1) \tag{60}$$

$$y = y + \Delta(2) \tag{61}$$

As the combination of the steepest descent and second-order gradient algorithms, the LM algorithm switches between the two algorithms during the least-squares minimization process. When the combination coefficient μ is very small (nearly zero), (59) approaches the second-order algorithm; when the combination coefficient μ is very large, (59) approaches the steepest descent method.

B. Density-Concentration process in the proposed DCMDS algorithm

Density based clustering methods assume that cluster centers are surrounded by neighbors with lower local densities. In other words, data with smaller local densities should move close to data with larger local densities when embedding. The general new idea behind this is the density concentration process, where each data will be mapped closer to its nearest neighbor data in the density field.

Step 1: Calculate the data local density ρ_i using (49).

Step 2: Generate nearest neighbor map (NNM) in the meantime of calculating the minimum distance δ (7). Nearest neighbor information will be stored in NNM, where data connects to its nearest neighbor of larger local density, with their distance equal to δ. For instance, if the minimum distance δ for data point m is found to be δ_m from data point m to data point n, then data point n is the nearest neighbor of data point m in the density field. Notice that data's nearest neighbor always has a larger local density, with their distance as δ. The datum with the largest local density connects to itself in the NNM.

$$\text{Moving rate: } \lambda = 1 - \frac{\delta(i)}{\max(\delta)} * \frac{\rho(i)}{\max(\rho)} \tag{62}$$

If λ is larger, the data will move more to its nearest neighbor in the density field but might not in the distance field. The embedded data $P^{(t)}$ in the t-th iteration after density concentration process will be as follows.

$$P^{(t)} = P^{(t-1)} + \lambda * (P_{NN} - P^{(t-1)}) \tag{63}$$

C. DCMDS algorithm

Although it is illustrated in this chapter that the LM method is capable of providing a better visualization than traditional MDS techniques, LM alone is not enough to generate impressive MDS results. Fortunately, by combining LM with the benefit of density concentration, the proposed DCMDS algorithm can generate inspiring MDS results. Detailed DCMDS algorithm is presented in Algorithm 5-1.

Algorithm: Simple version of the proposed DCMDS
Purpose: MDS **Input**: Unlabeled raw data **Output**: Embedded data *P* in desired dimension (i.e. 2D) **Algorithm**: compute d; % between-data distances

compute ρ; % each datum's local density using (49);
generate NNM;
initial solution $P^{(0)}$ from matrix eigen-decomposition; % (56)
$\alpha = 0.5; \beta = 0.5;$ % $\alpha + \beta = 1$
for $t = 1$ **to** T **do** % T is the number of iterations
  compute $P^{(t)}$ using LM algorithm based on the previous solution $P^{(t-1)}$;
  compute moving rate λ; % (62)
  set $P^{(t)} = \alpha * P^{(t)} + \beta * (P^{(t-1)} + \lambda * (P_{NN} - P^{(t-1)}))$; (64)
    % P_{NN} is data's nearest neighbor found in NNM.
end

Algorithm 5-1 Simple version of the proposed DCMDS

5.3 Experimental Results

In this section, the proposed DCMDS algorithm is evaluated using experiments on several real-world data. All the experiments are implemented based on the same software and hardware: Matlab R2013a in OS X operating system with Intel Core i5 (I5-4258U) @2.4GHz 8.00GB memory.

5.3.1 Human Activity Recognition (HAR) Using Smartphone Dataset

Nowadays, more and more carry-on devices are invented to measure human activities, e.g. work out statistics etc. Visualization techniques can provide us a good knowledge of our activity record, such as HAR [70] data. HAR consists of 7,352 data with 561 dimensions/attributes, built from the recordings of thirty volunteers performing activities of daily living while carrying a waist-mounted smartphone with embedded inertial sensors that are in charge of collecting the raw data information. These points belong to one of the following six categories: C1-walking, C2-walking upstairs, C3-walking downstairs, C4-sitting, C5-standing, and C6-laying.

Figure 5.1-Figure 5.5 show the experiment results for Sammon mapping, Isomap, LLE, LAMP and the proposed DCMDS, respectively, on the HAR dataset. In Figure 5.1, Sammon mapping shows two distinct clusters with overlapping classes in each cluster. In Figure 5.2, Isomap shows two distinct, dense clusters with overlapping classes of even less distinction

compared to Figure 5.1. LLE in Figure 5.3 shows nearly no distinction among classes outside of being in the left or right loosely collected clusters. Similar to Sammon mapping, LAMP in Figure 5.4 shows two distinct clusters with little distinction between classes within each cluster.

In contrast, DCMDS in Figure 5.5 shows distinct clusters with better defined class separation within each cluster. Whereas some of the competing methods show sharp class distinction for no more than a few classes, DCMDS shows distinct separation in almost every part of the map, save the two overlapping classes designated in pink and yellow. The optimal performance of DCMDS for this dataset is verified in a later performance evaluation.

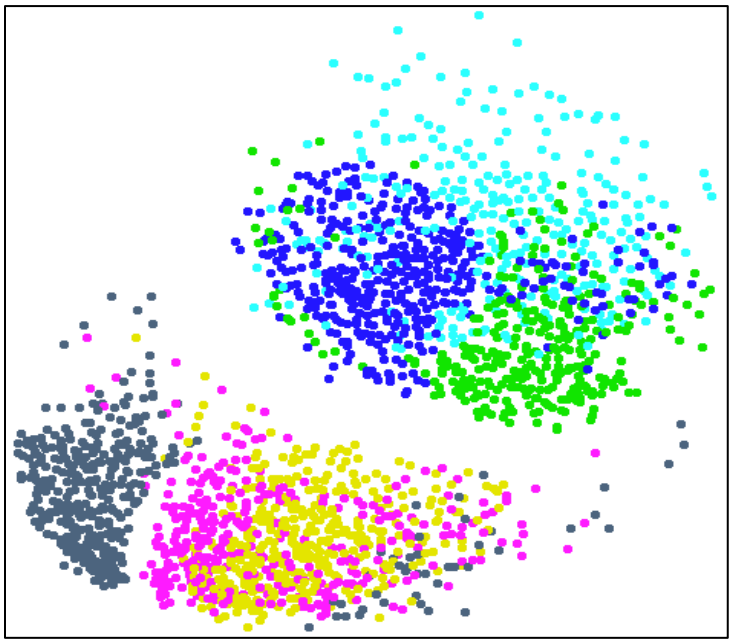

Figure 5.1 Sammon mapping on HAR. (time=88.77 sec)

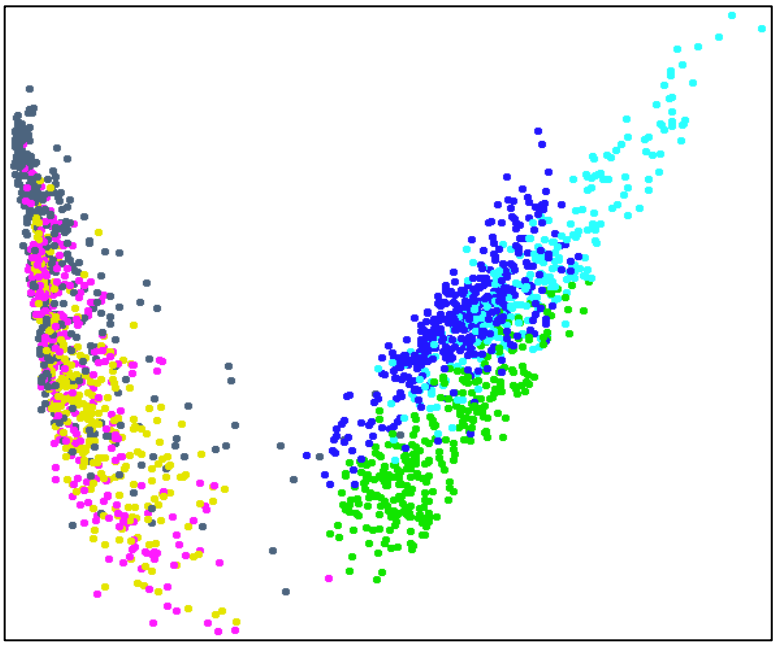

Figure 5.2 Isomap on HAR. (time=34.13 sec)

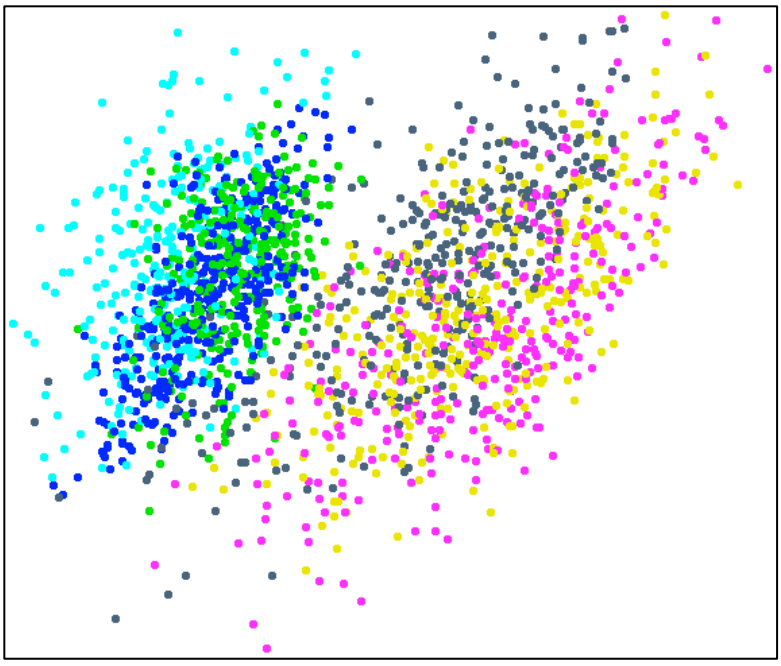

Figure 5.3 LLE on HAR. (time=1.39 sec)

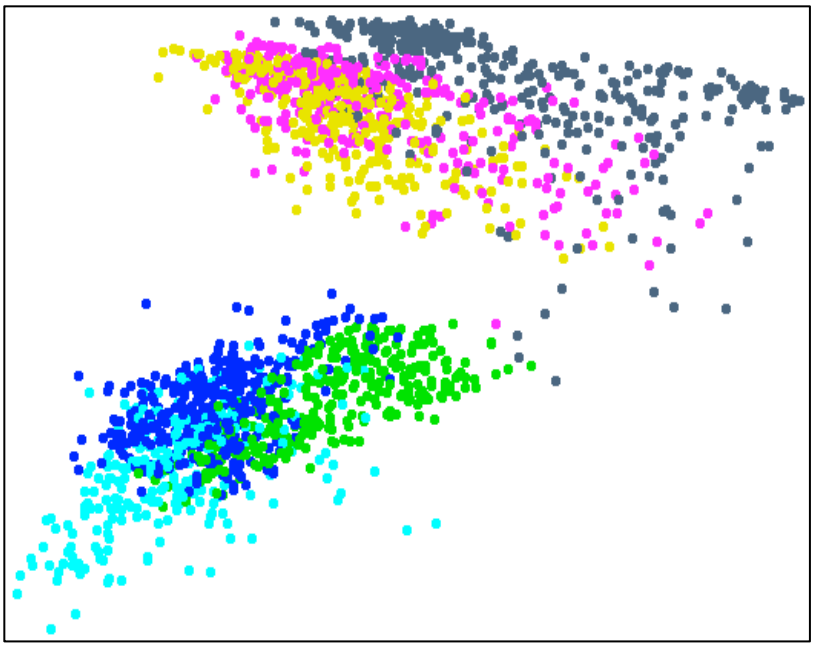

Figure 5.4 LAMP on HAR. (time= 9.73 sec)

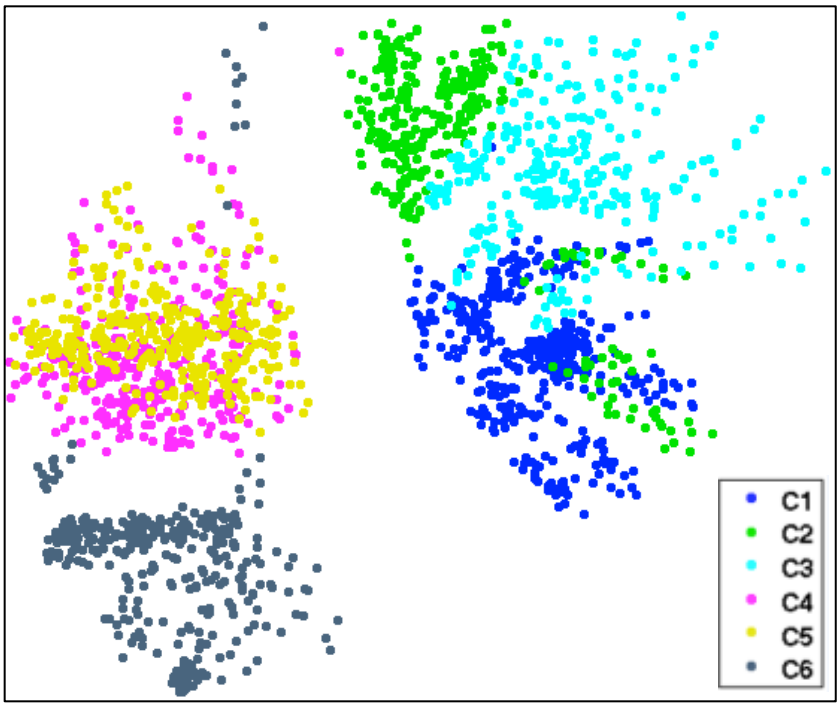

Figure 5.5 DCMDS on HAR. (time= 27.21 sec).

5.3.2 MNIST Handwritten Digits Dataset

MINST handwritten digits [71] recognitions have been considered as one of the most complex and difficult problems to be solved. It consists of 60,000 photos with the size of 28*28 (or 784 dimensions).

Figure 5.6-Figure 5.10 contain experiment results on first 6,000 photos of the MNIST dataset using Sammon mapping, Isomap, LLE, LAMP and the proposed DCMDS, respectively. In Figure 5.6, Sammon mapping shows all data is contained in a large circle with many overlapping classes due to being trapped in local minima, demonstrating poor visualization; this technique is only able to distinguish a single class. Figure 5.7 shows that Isomap is able to distinguish the same class as Sammon mapping in a more compact, distinctive manner, yet struggles to show good visualization on the remaining overlapping classes. LLE visualization in Figure 5.8 demonstrates better visualization than the former methods with a tight, distinct class clusters on the edges, but still has little distinction in the center. LAMP visualization also has small distinction in the edges, with most of the data barely distinguishable in the center. Unlike the other methods, the proposed DCMDS in Figure 5.10 shows distinct classes with significantly less overlap with visual separation between clusters. Similar to the HAR dataset, the proposed DCMDS's superiority is later verified in a performance evaluation.

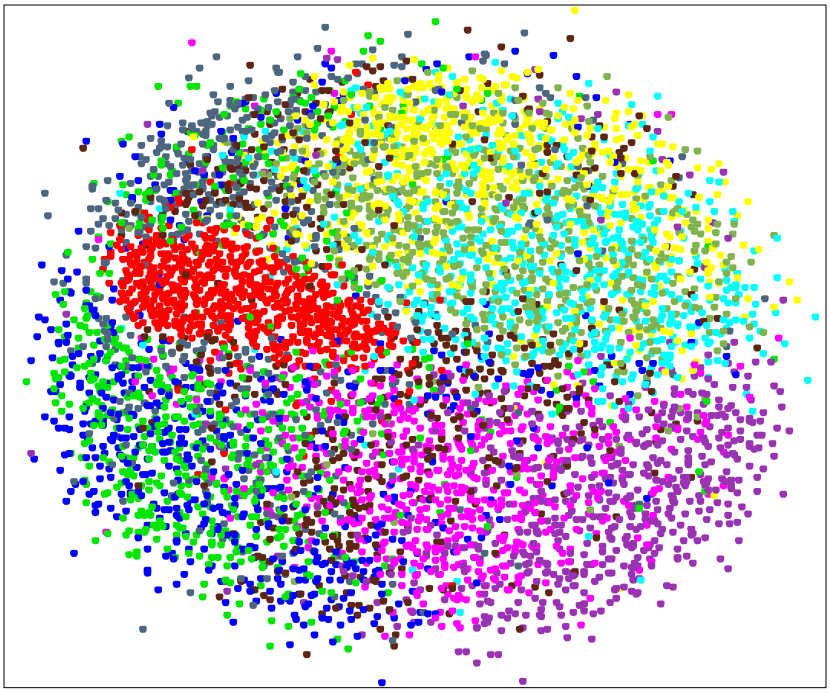

Figure 5.6 Sammon mapping on MNIST. (time=11,362.07 sec)

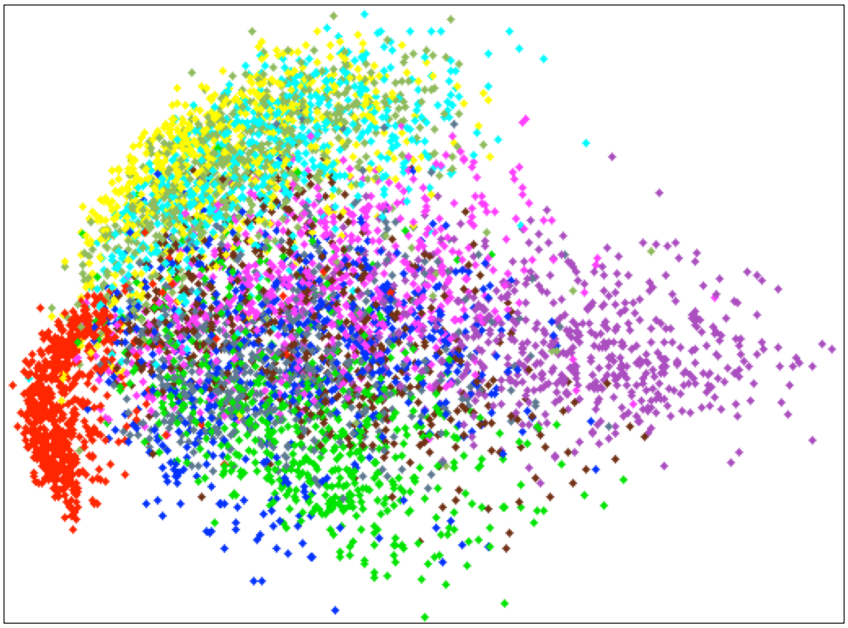

Figure 5.7 Isomap on MNIST. (time=163.25 sec)

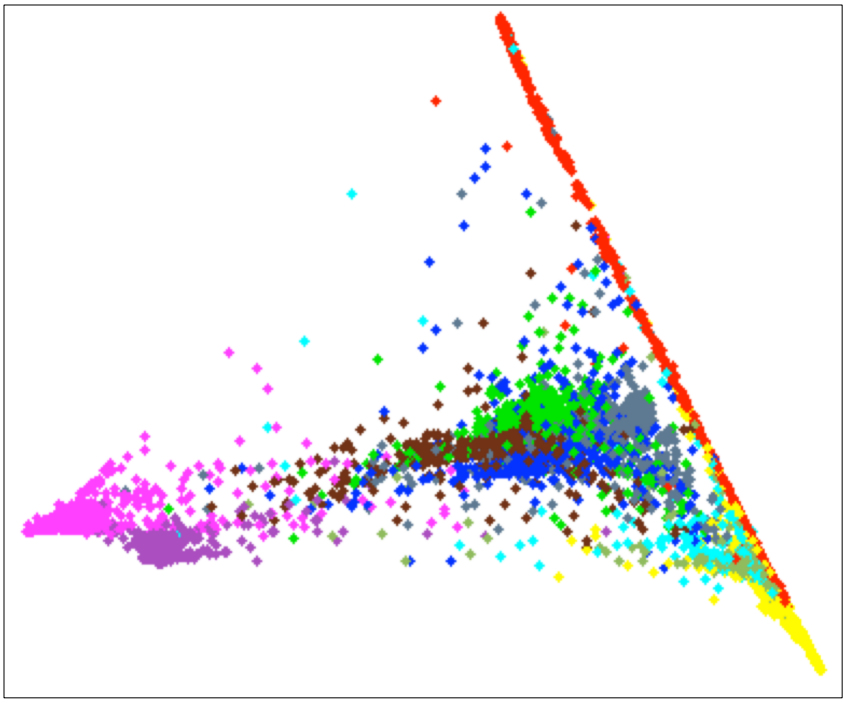

Figure 5.8 LLE on MNIST. (time=29.65 sec)

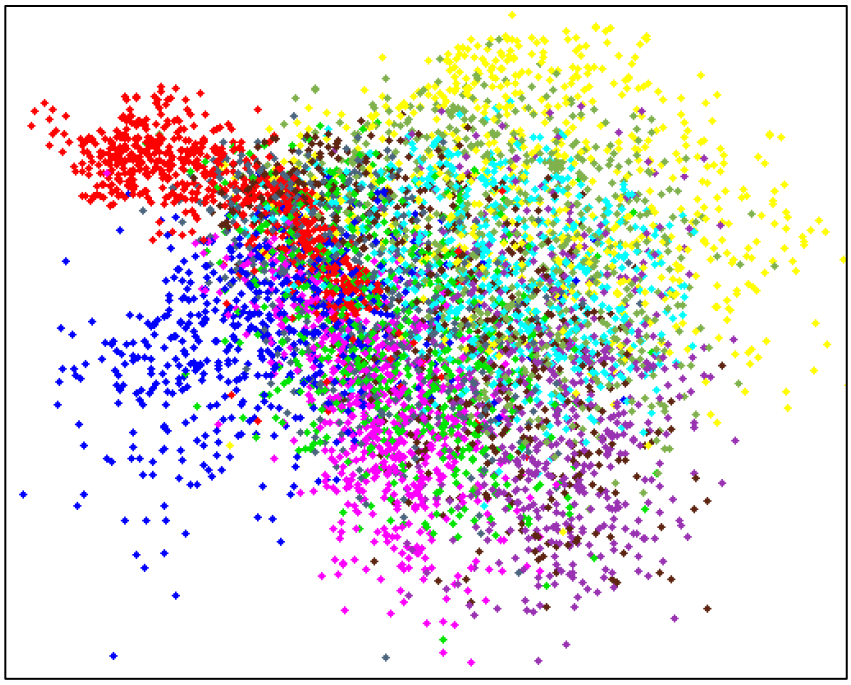

Figure 5.9 LAMP on MNIST. (time=67.92 sec)

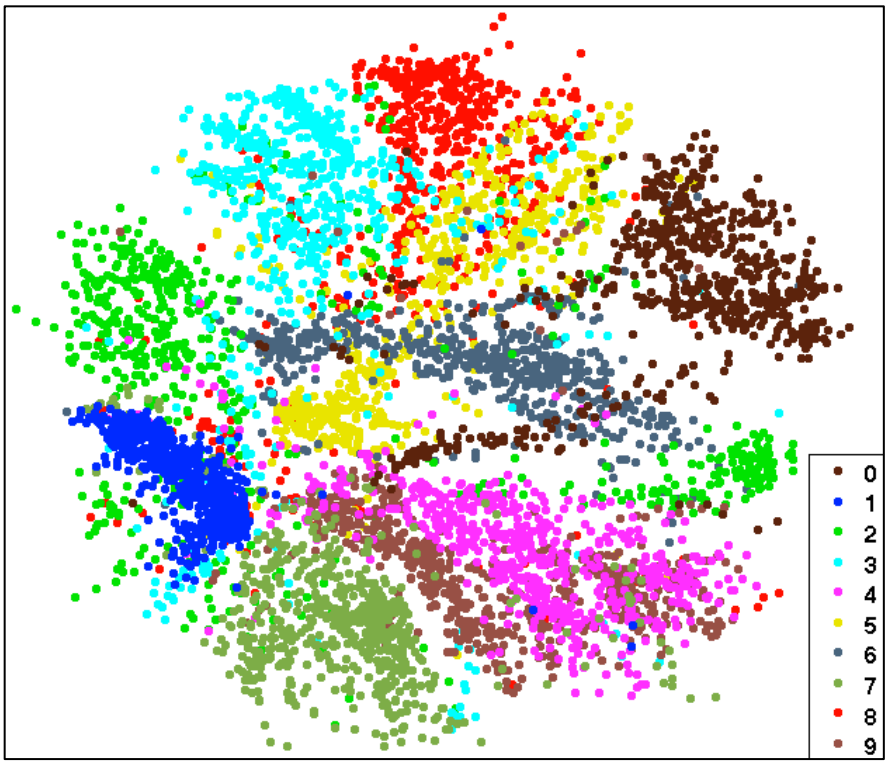

Figure 5.10 DCMDS on MNIST. (time= 1,283.85 sec)

5.3.3 Olivetti Face Dataset

Olivetti face data [72] is a set of 112*92 (or 10,304 dimensions) images of different persons with different face angels, expressions and even with or without glasses wearing. In Figure 5.12, Sammon mapping shows relatively strong visualization results, with some overlapping classes seen throughout the map. Isomap, LLE, and LAMP in Figure 5.13-Figure 5.15 all fail to provide distinct class separation. In contrast, the DCMDS results in Figure 5.16 provide significant class distinction, far outperforming all other methods. Only a single datum is incorrectly mapped. The Olivetti face dataset clearly shows great performance of the DCMDS algorithm compared to the competing methods.

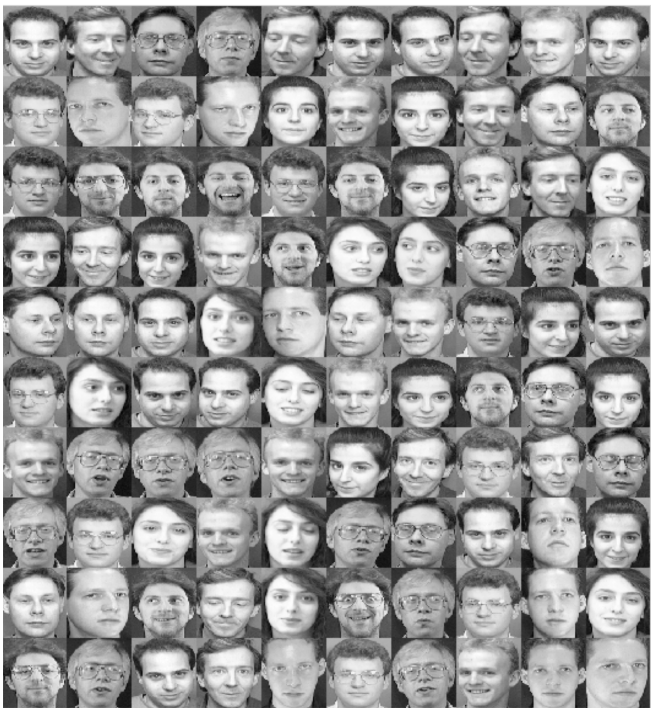

Figure 5.11 Original Olivetti face images. (100 samples)

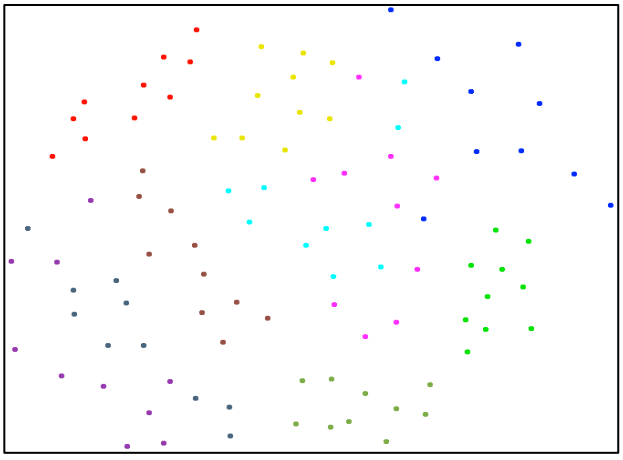

Figure 5.12 Sammon mapping on Olivetti Faces. (time=1.04 sec)

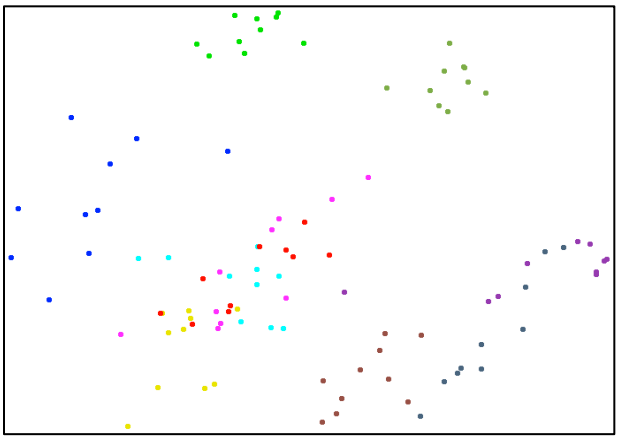

Figure 5.13 Isomap on Olivetti Faces. (time=0.13 sec)

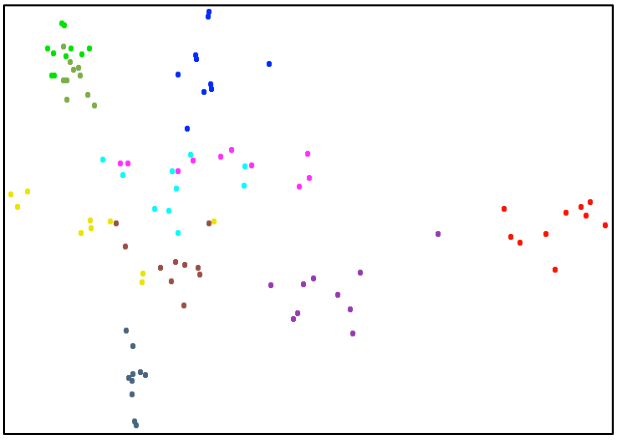

Figure 5.14 LLE on Olivetti Faces. (time=0.09 sec)

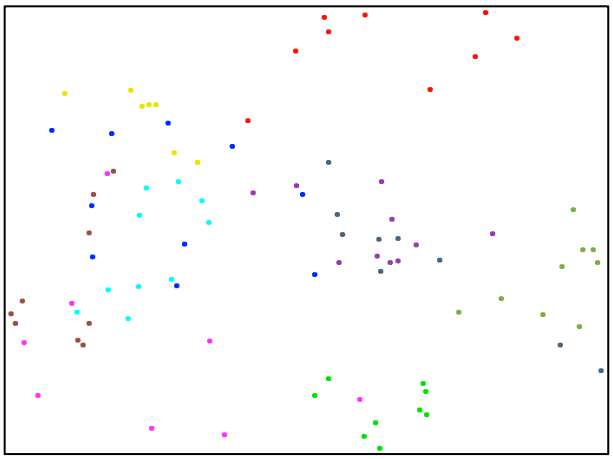

Figure 5.15 LAMP on Olivetti Faces. (time=0.12 sec)

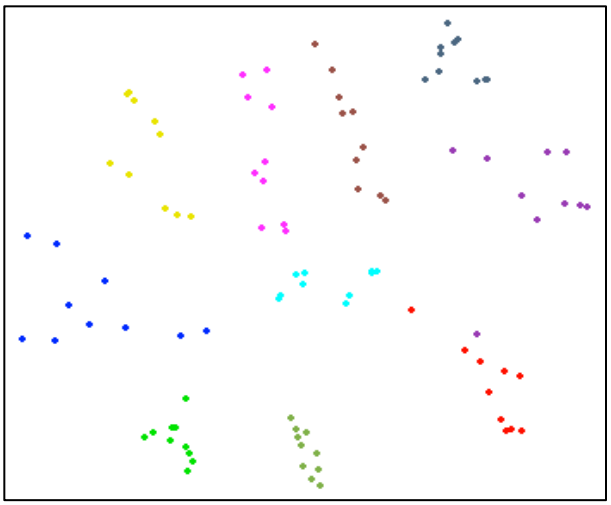

Figure 5.16 DCMDS on Olivetti Faces. (time=1.09 sec)

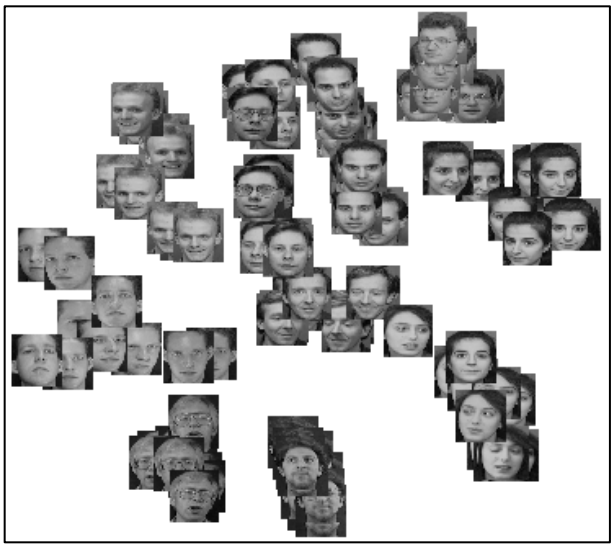

Figure 5.17 DCMDS with Olivetti face photos.

5.3.4 Performance Evaluation

The concept of using a loss function to evaluate performance of MDS came with J.B. Kruskal [9] and gave us the concept of minimizing a loss function called stress. The disparity in stress is a measure of how well the Euclidean distance in low-dimensional space matches the dissimilarity, which is usually the Euclidean distance in high-dimensional space.

It's proven in the original paper that the order of the original dissimilarities is preserved by the disparities. A loss function L, which is really stress, is defined as follows

$$L = stress = \sqrt{\frac{\sum_{r<s}(d_{rs}-\hat{d}_{rs})^2}{\sum_{r<s} {d_{rs}}^2}} \tag{65}$$

where d_{rs} is the Euclidean distance in low-dimensional space between point r and s. $\hat{d}_{rs}$ is the disparity corresponding to d_{rs}. The order that indicates $\{r < s\}$ is determined by the Euclidean distance relationships in high-dimensional space. When the MDS map perfectly reproduces the input data, $d_{rs} - \hat{d}_{rs}$ is zero for all r and s, so stress is zero. Thus, generally speaking, the smaller the stress, the better the representation. A detailed Kruskal stress factors using different MDS approaches for different datasets are listed in Table 5-1.

Table 5-1 Comparisions of Kruskal stress on experimental data

	HAR (N_p=2,000)	MNIST (N_p=6,000)	Olivetti Face (N_p=100)
Sammon mapping	s = 0.1481	s = 0.4320	s = 0.2678
Isomap	s = 0.2254	s = 0.3930	s = 0.3123
LLE	s = 0.6750	s = 0.7896	s = 0.6187
LAMP	s = 0.1566	s = 0.4051	s = 0.3134
DCMDS	s = **0.1422**	s = **0.3408**	s = **0.2563**

(N_p: test data points number)

The results clearly demonstrate a consistent superior performance of the DCMDS approach. Although other methods may have positive results (but still worse than DCMDS) on some of the datasets, our method succeeds each time. As the iterative methods are not advanced in processing time, e.g. Sammon mapping and the DCMDS methods, our method is not favorable in fast mapping techniques. However, the Kruskal stress factor results, in combination with the ability to produce visual class distinctions, favor the DCMDS approach as a highly successful data visualization method.

5.4 Conclusion

The major innovation of the new algorithm is the incorporation of density concentration into our improved MDS with LM optimization. The DCMDS algorithm can reveal both the distance relationships as well as the density relationships between data categories, and therefore successfully provide us a better visualization on data. Experimental figures also give us vivid visualization results. Compared with other state-of-the-art MDS methods, our proposed DCMDS achieve the best performance. More importantly, DCMDS provides an integrated density based MDS approach to perform small cluster formations and embedding simultaneously without any prior knowledge, which can be easily applied to project data into any desired space. Besides, the proposed DCMDS algorithm can be very useful and heuristic for combining unsupervised clustering methods with traditional MDS techniques in further study.

In summary, the proposed DCMDS algorithm has several important merits as follows:

1) General-MDS-purpose. DCMDS algorithm applies LM method to find the embedded locations for data based on their mutual distances and therefore is a general-purpose MDS approach. Thus, it is suitable to conduct dimensionality reduction, visualization， and other purposes that general MDS approaches are used for. Besides, it is capable of escaping "co-located" problems that appear in traditional MDS techniques.

2) Microclusters forming. Density based clustering methodology (an unsupervised technique) is used to concentrate clusters so that the margins between clusters are enlarged. As clusters are formed and separations between clusters expand, a better visualization of microclusters is expected.

3) Absence of parameter setting/integrated. Simple and efficient, the DCMDS algorithm integrates cluster concentration based on local densities with MDS approach based on LM

optimization by linear combination set (64). It is easy to be interpreted and fulfilled. Most importantly, it is parameter-setting free and unsupervised.

CHAPTER 6 VISUALIZING RELATIONS BETWEEN DATA

In the last chapter, we introduced our DCMDS algorithm to visualize high-dimensional data in a low-dimensional space. It is capable to generate satisfying mappings with better separation between microclusters and better concentration in clusters. Beyond that, if we can reveal the relations between data during visualization process, it can be even cheerful. As this world connected more and more tight by all sorts of data, it's far more than valuable to analyze the relations between data. In this chapter, we will extend our DCMDS algorithm to a degree that it has the capability to show the relations between data. Experiments on real world data are conducted to verify our attempts.

The rest of this chapter is organized as follows. Section 6.2 introduces the nearest neighbor map, which contains data relation information. Section 6.3 presents our proposed DCMDS-RV algorithm for data relations visualization. Section 6.4 demonstrates the experimental results on several high-dimensional datasets to evaluate the effectiveness of the proposed algorithm. Finally, Section draws the conclusions.

6.1 Introduction

Except for data dimension reduction, our goal is to visualize complex high-dimensional data as networks, or graphs in low-dimensional space. As mentioned in Section 5.1, traditional MDS methods still have one more shortcoming:

1) Connected relations are not shown in the embedded results.

In this chapter, we revisited our DCMDS algorithm and asked the question: *Can we reveal relations between data in the resulted embedding?* The answer is definitely yes because there are many relations we can use, e.g. neighbors, nearest neighbors etc. Therefore, in order to

utilize our DCMDS algorithm to fulfill relation visualization purpose, we propose a Density-Concentrated Multi-Dimensional Scaling algorithm for relations visualization, called DCMDS-RV. The key idea behind the proposed DCMDS-RV algorithm is to use density based clustering in DCMDS to generate the nearest neighbor map and show the map in DCMDS results. Other than the merits of DCMDS, the main contribution of DCMDS-RV is summarized as follows:

In order to reveal the relations between data, nearest neighbor map (NNM) is generated along DCMDS algorithm and no extra step is needed. The relations between data are shown in the embedded results as connected lines, providing a more vivid and intuitive view of data.

6.2 Relations between Data

Before showing relations between data in the DCMDS results, we should think about what kind of relations/connections we want to be visualized. Based on the common notation that if two data points are connected in a graph, they are thought to be related and, more specifically, similar. As neighboring information contained the data similarities, it is preferable to reveal the nearest neighbor relations in the embedding results.

6.2.1 Density Based Clustering

In our DCMDS-RV algorithm, the utilized density based clustering follows the same strategy shown in Section 5.2.1.

6.2.2 Relations Based on Data Density in Density Based Clustering

Density based clustering methods assume that cluster centers are surrounded by neighbors with lower local densities. In other words, data with smaller local densities should move close to data with larger local densities when embedding. The general new idea behind this is the density concentration process, where each data will be mapped closer to its nearest neighbor data in the density field. Two steps are needed in generating nearest neighbor map:

Step 1: Calculate the data local density ρ_i using (49).

Step 2: Generate nearest neighbor map (NNM) in the meantime of calculating the minimum distance δ (7). Nearest neighbor information will be stored in NNM, where data connects to its nearest neighbor of larger local density, with their distance equal to δ. For instance, if the minimum distance δ for data point m is found to be δ_m from data point *m* to data point *n*, then data point *n* is the nearest neighbor of data point m in the density field. Notice that data's nearest neighbor in NNM always has a larger local density, with their distance as δ. The datum with the largest local density connects to itself in the NNM. This also generates the relations between data in the density field, which are shown connected in the resulted embedding. A simple example of NNM for a two-dimensional dataset is shown in Figure 6.1.

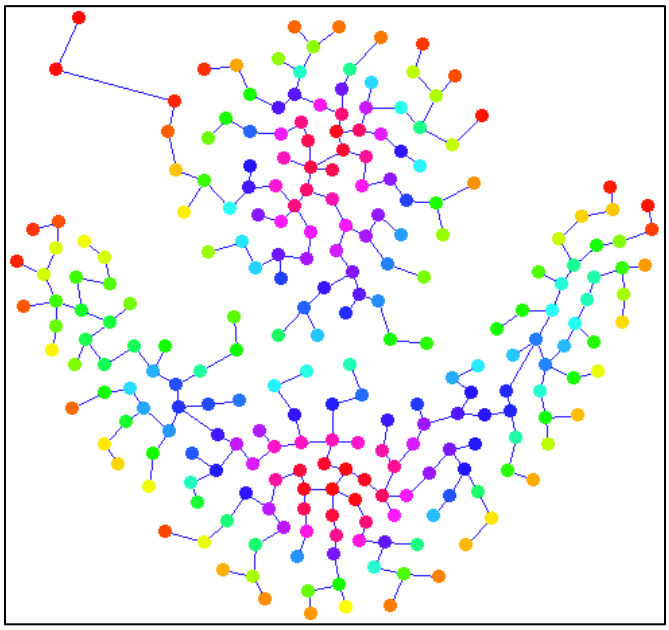

Figure 6.1 An illustration example of NNM for two-dimensional dataset-FLAME [39]. The colors of data stand for different density values. In NNM, data's nearest neighbor always has a larger local density, with their distance as δ.

6.3 Visualizing Relations between Data

There will be one more step compared with DCMDS in our DCMDS-RV method. The detailed algorithm is illustrated as follows.

Algorithm: DCMDS-RV

Purpose: MDS for Relations Visualization
Input: Unlabeled raw data
Output: Embedded data P in desired dimension (i.e. 2D) with relations revealed
Algorithm:
compute d; % between-data distances
compute ρ; % each datum's local density using (49);
generate NNM;
initial solution $P^{(0)}$ from matrix eigen-decomposition; % (56)
$\alpha = 0.5; \beta = 0.5;$ % $\alpha + \beta = 1$
for $t = 1$ **to** T **do** % T is the number of iterations
compute $P^{(t)}$ using LM algorithm based on the previous solution $P^{(t-1)}$;
compute moving rate λ; % (62)
set $P^{(t)} = \alpha * P^{(t)} + \beta * (P^{(t-1)} + \lambda * (P_{NN} - P^{(t-1)}))$; (66)
% P_{NN} is data's nearest neighbor found in NNM.
end
connect each datum to its nearest neighbor in NNM.

Algorithm 6-1 DCMDS-RV algorithm.

6.4 Experimental Results

Several real-life datasets are illustrated in this section to demonstrate the effectiveness of proposed DCMDS-RV algorithm. All the experiments are implemented based on the same software and hardware: Matlab R2013a in OS X operating system with Intel Core i5 (I5-4258U) @2.4GHz 8.00GB memory.

6.4.1 Human Activity Recognition (HAR) Using Smartphone Dataset

Data generated and recorded by smart devices are growing explosively and becoming more and more important to analyze our behaviors. HAR [70] consists of 7,352 data with 561 dimensions/attributes, built from the recordings of thirty volunteers performing activities of daily living while carrying a waist-mounted smartphone with embedded inertial sensors that are in charge of collecting the raw data information. These points belong to one of the following six

categories: C1-walking, C2-walking upstairs, C3-walking downstairs, C4-sitting, C5-standing, and C6-laying.

Figure 5.1 to Figure 5.4 show the experiment results for Sammon mapping, Isomap, LLE, and LAMP, respectively, on the HAR dataset. In these figures, little distinction or no distinctions are shown in clusters.

In contrast, DCMDS-RV in Figure 6.2 shows distinct clusters with better defined class separation within each cluster. What's more, the relations between data are clearly shown as well, helping to better understand data topology. Whereas some of the competing methods show sharp class distinction for no more than a few classes, DCMDS-RV shows distinct separation in almost every part of the map, save the two overlapping classes designated in pink and yellow. The optimal performance of DCMDS-RV for this dataset is verified in a later performance evaluation.

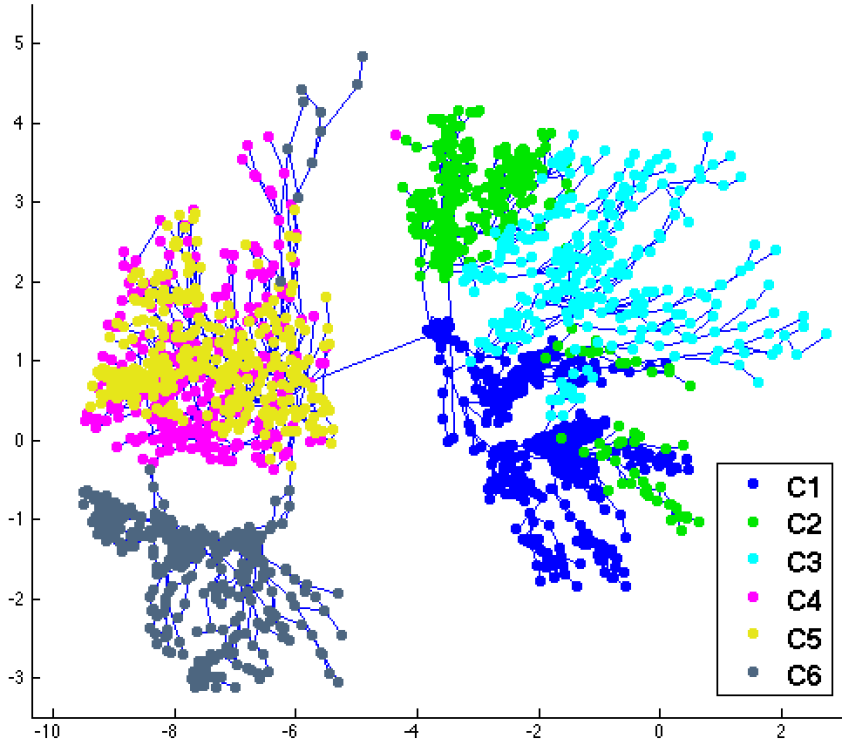

Figure 6.2 DCMDS-RV on HAR. (time= 27.21 sec).

6.4.2 MNIST Handwritten Digits Dataset

MINST handwritten digits [71] recognitions have been considered as one of the most complex and difficult problems to be solved. It consists of 60,000 photos with the size of 28*28 pixels (784 dimensions). Sample MNIST digits are shown in Figure 6.3.

Figure 5.6-Figure 5.10 contain experiment results on first 6,000 photos of the MNIST dataset using Sammon mapping, Isomap, LLE, and LAMP, respectively. In these figures, it's hard to distinguish one cluster from other clusters. Unlike the other methods, the proposed DCMDS-RV in Figure 6.5 shows distinct classes with significantly less overlap with visual separation between clusters. Besides, the NNM is revealed in Figure 6.4 so that the relations between data can be favorably visualized (Figure 6.5). Figure 5.10 shows the resulted figure without showing the relations. Similar to the HAR dataset, the proposed DCMDS-RV's superiority is later verified in a performance evaluation.

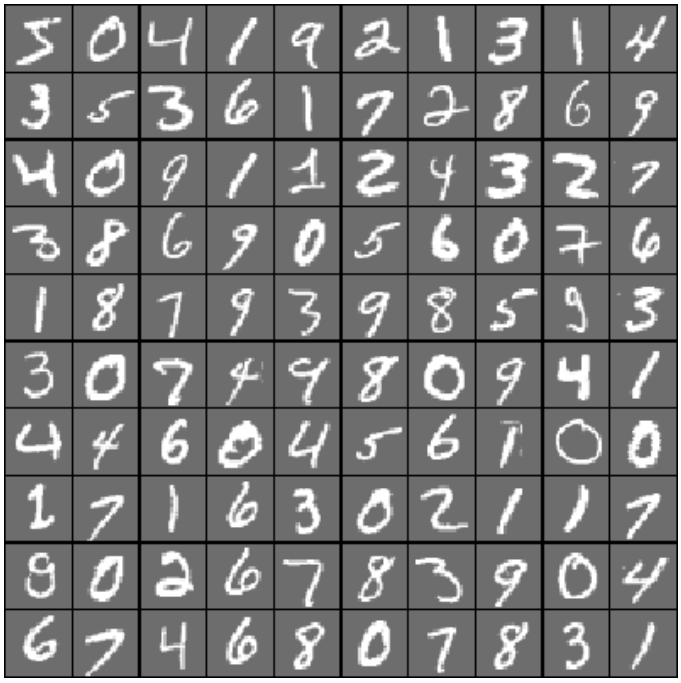

Figure 6.3 Sample MNIST data (first 100 digits).

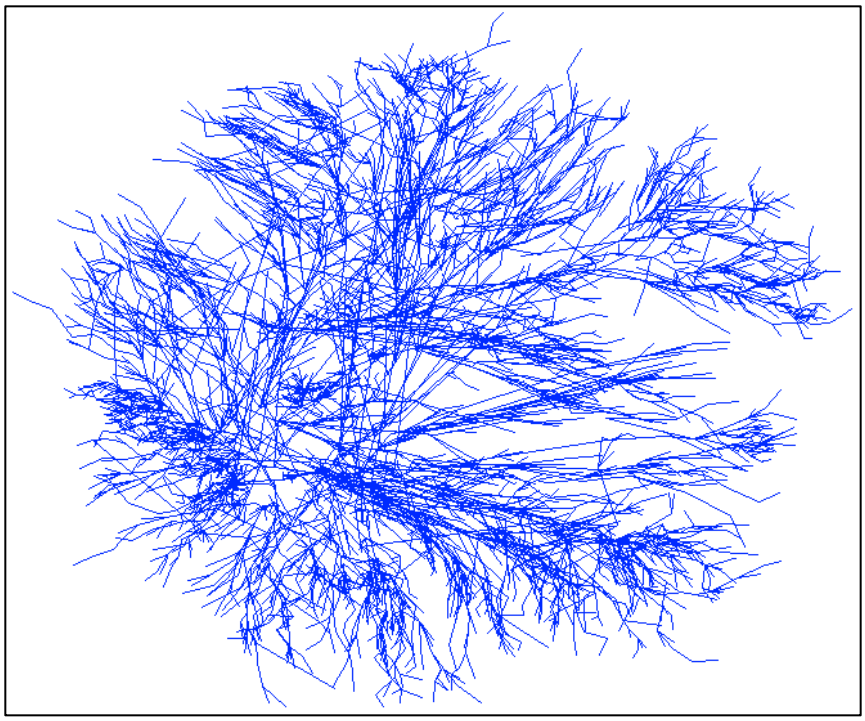

Figure 6.4 NNM for MNIST using DCMDS-RV.

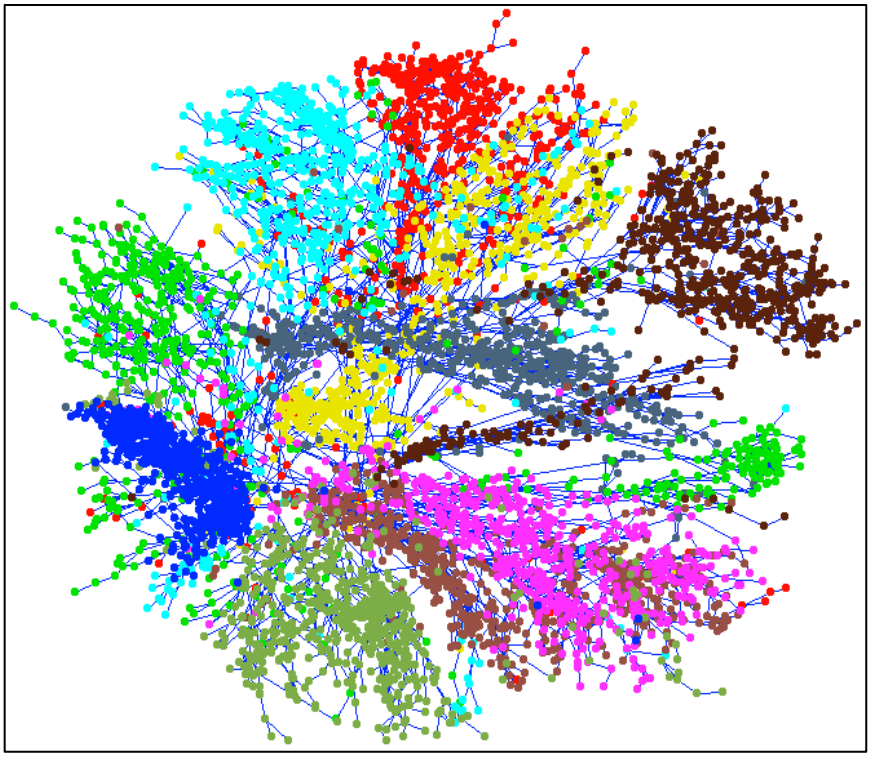

Figure 6.5 DCMDS-RV on MNIST. (time= 1,283.85 sec)

6.4.3 Olivetti Face Dataset

Olivetti face data [72] is a set of 112*92 (or 10,304 dimensions) images of different persons with different face angels, expressions and even with or without glasses wearing, as Figure 6.6 shows. In Figure 5.12, Sammon mapping shows relatively strong visualization results, with some overlapping classes seen throughout the map. Isomap, LLE, and LAMP in Figure 5.13-Figure 5.15 all fail to provide distinct class separation. In contrast, the DCMDS-RV results in Figure 6.7 provide significant class distinction with each datum collected to another, far outperforming all other methods. Face data are favorably viewed in a graph, which looks like tree branches. Only a single datum is incorrectly related. The Olivetti face dataset clearly shows great performance of the DCMDS-RV algorithm compared to the competing methods. Face images (Figure 6.6) are eventually categorized into classes using the presented approach, shown in Figure 6.8.

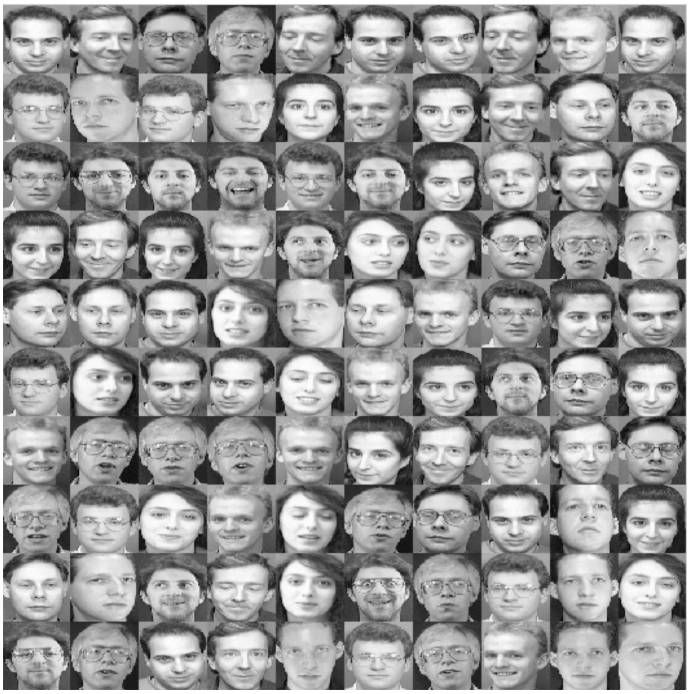

Figure 6.6 Original Olivetti face images. (100 samples)

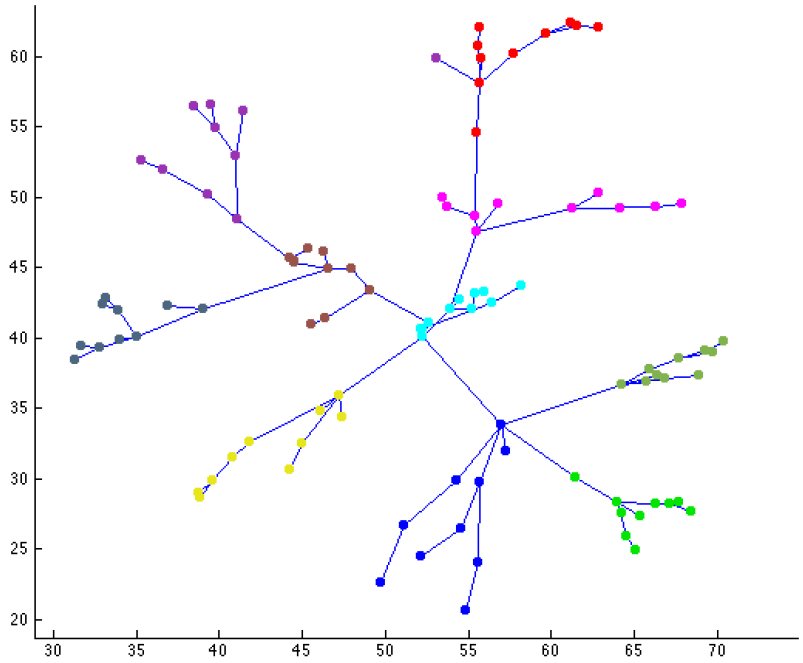

Figure 6.7 DCMDS-RV on Olivetti Faces. (time=1.09 sec)

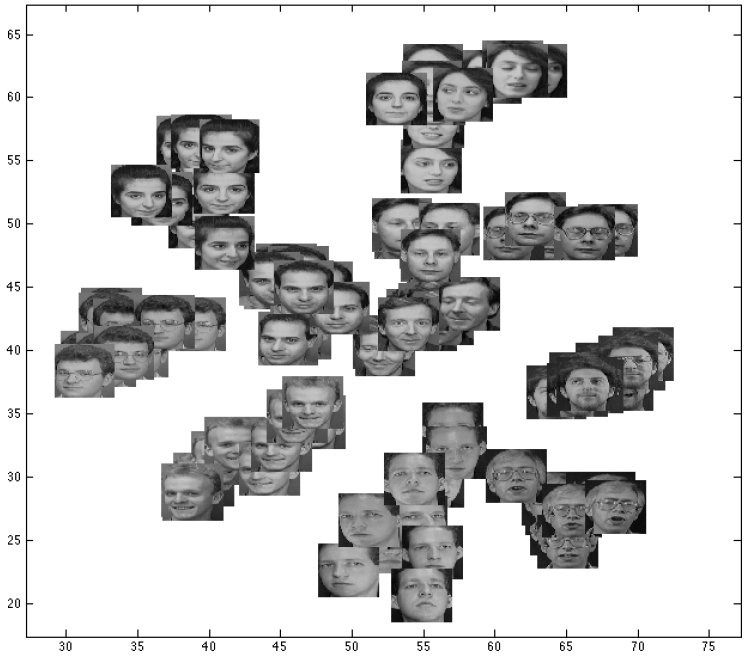

Figure 6.8 Corresponding face images of Figure 6.7.

6.4.4 Wikipedia 2014 Words Dataset (Text Relation Visualization)

Analysis on text such as text relation extraction reveals useful information. Here, wikipedia corpuses are visualized using the proposed approach. Wikipedia corpuses [73] contain the full text of Wikipedia, and they contain 1.9 billion words in more than 4.4 million articles. It allows you to search Wikipedia in a much more powerful way than is possible with the standard interface. You can search by word, phrase, part of speech, and synonyms. It relates to microbiology, economics, basketball, Buddhism, or thousands of other topics. Wikipedia 2014 words dataset is the Wikipedia corpuses of year 2014, with a vocabulary of the top 200,000 most frequent words. In the result of Figure 6.9, the proposed approach provides an impressive visualization on text. The graph topology is successfully shown between text. Four zoomed-in regions in the resulted graph are clearly illustrated in Figure 6.10, where related words are connected as topology.

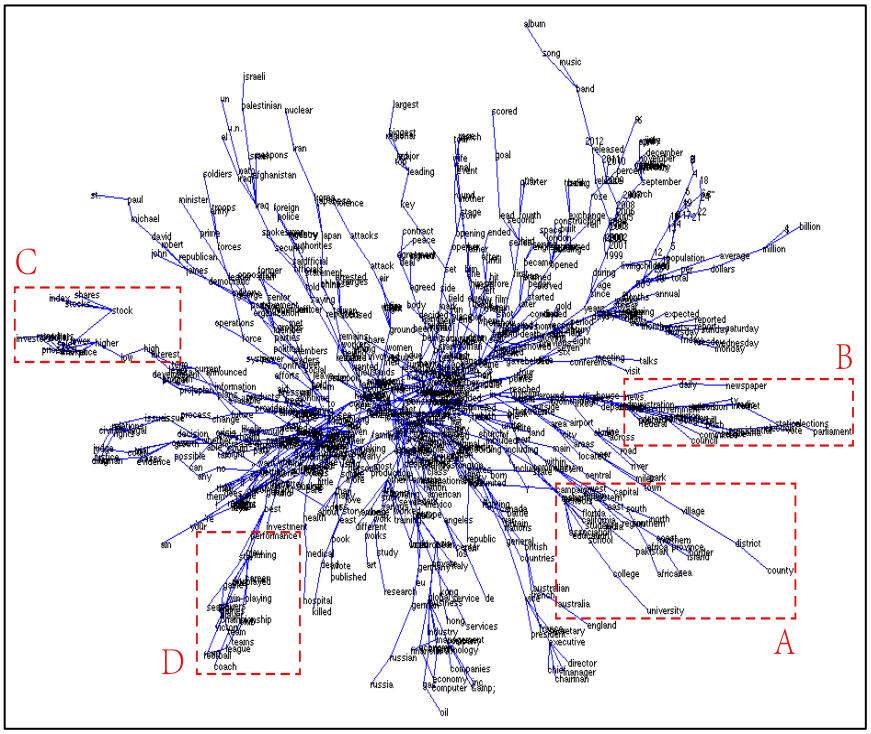

Figure 6.9 DCMDS-RV on Wikipedia 2014 Words (956 samples).

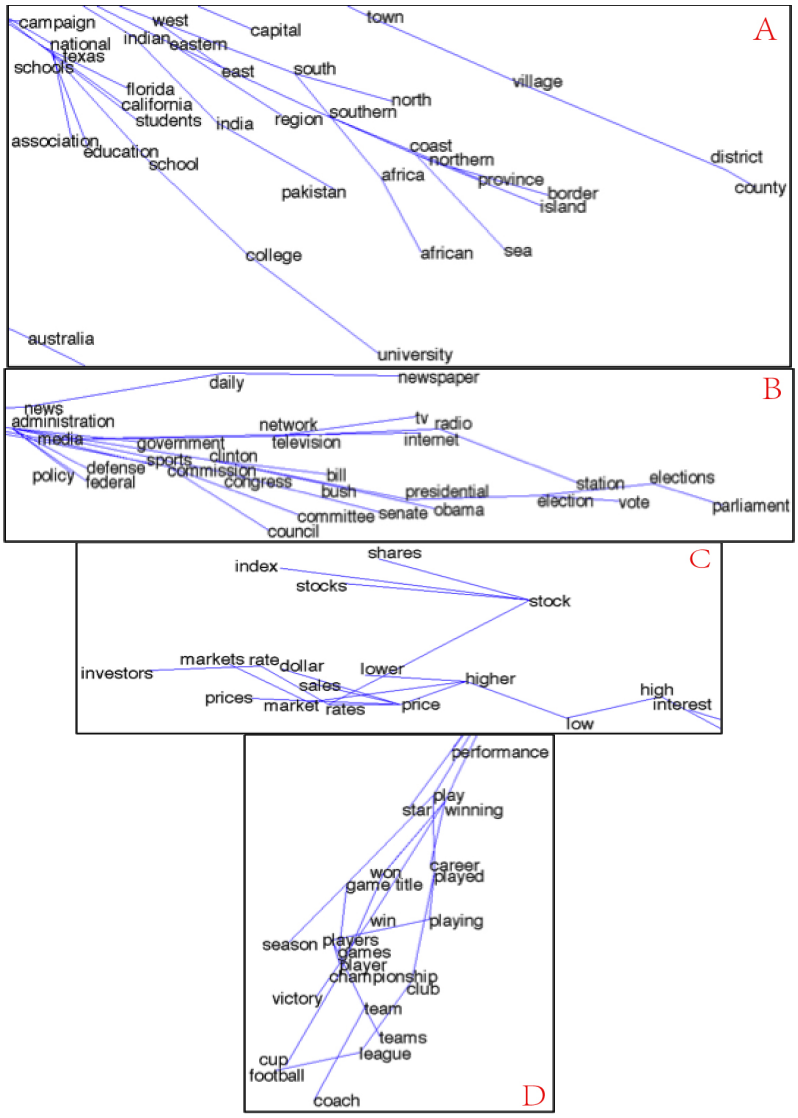

Figure 6.10 Zoomed-in visualization results in the corresponding marked region in Figure 6.9.

6.4.5 Performance Evaluation

The Kruskal stress factor (Section 5.3.4) is used to evaluate the mentioned MDS techniques. A detailed Kruskal stress factors using different MDS approaches for different datasets are shown in Figure 6.11.

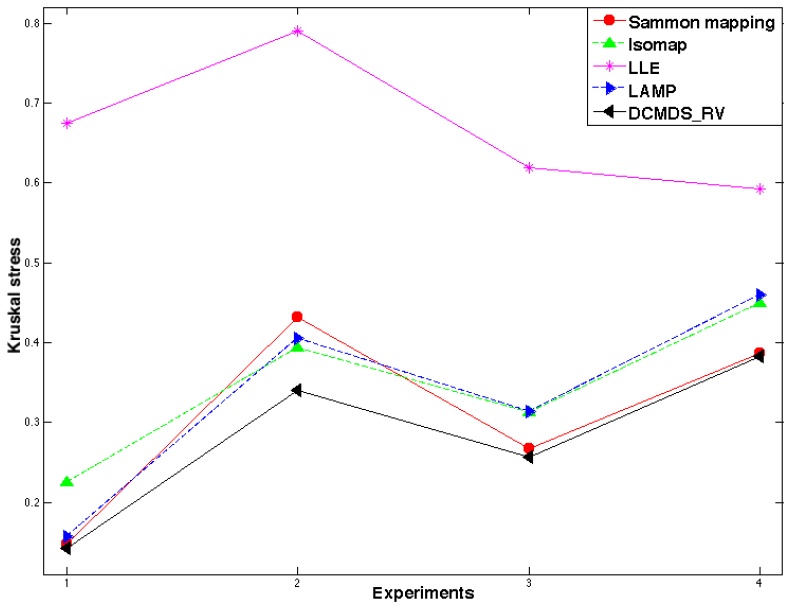

Figure 6.11 Kruskal stress factors for the four experiments using Sammon mapping, Isomap, LLE, LAMP, and DCMDS-RV. Experiments 1: HAR; 2: MNIST; 3: Olivetti Face; 4: Wikipedia 2014 Words.

6.5 Conclusion

This chapter presents a new MDS algorithm called DCMDS-RV for data relation visualization in a low-dimensional space. It takes advantage of DCMDS algorithm and can reveal both the distance relationships as well as the density relationships between data, and therefore successfully provide us a better visualization on all sorts of high-dimensional data as graph topology. NNMs are generated along the algorithm, which can provide data nearest neighbor information in density field, showing how data are connected to the others. Unlike the traditional MDS approaches with only one purpose of embedding data, the proposed DCMDS-RV algorithm is capable of showing how the data are related/connected to the others. With the connections between related data shown, more information can be revealed in the embedded results. Experimental figures also give us vivid visualization results with relations shown.

Compared with other state-of-the-art MDS methods, our proposed DCMDS-RV achieves better performance.

CHAPTER 7 CONCLUSION AND FUTURE WORK

In this work, three clustering methods including the clustering method based on grid and density peaks, fast density and grid based clustering method and the clustering method by analyzing density and minimum internal and external distance ratios have been addressed to solve the data clustering tasks with complex shapes and noise. Moreover, visualization technique with clustering purpose as the auxiliary methodology has been introduced to fulfill Multi-Dimensional Scaling tasks and data relations visualization. These works are summarized in this chapter, and furthermore, the suggestions for future work are also provided.

7.1 Summary of Research

Throughout this work, all the objectives shown in Section 1.4 have been well accomplished. To be specific, the fulfillments are as follows:

- Accelerated data clustering.

Clustering method by finding density peaks has the potential capability to deal with data with complex shapes and various dimensions. However, it is computationally expensive. A grid-based clustering method by finding density peaks is proposed, which is fast and has the computation complexity of $O(n * grid\_size)$. It introduces the fuzzy assignment to compute grid node's density. Experimental results show that it is practical to classify data (even with noise) into different categories. In terms of efficiency and effectiveness, experiments on real-world datasets show that the proposed method significantly outperforms the original clustering method by finding density peaks in the process of calculating the local densities and assigning data points into different categories due to the use of the standard grid and sparse matrix technique respectively.

- Capable of clustering data that have complex shapes and noise.

Grid based clustering algorithms are favored for their processing speed. A density and grid based clustering method is developed, which takes advantage of grid based clustering. It is designed for the task of clustering data with arbitrary shapes and noise, which most of the other clustering approaches are not capable of. A novel strategy of finding mountain ridges as the outlook of a cluster is explained. Moreover, specific classification of border patterns and noises is discussed as well. Therefore, instead of giving a specific desired number of clusters, the presented density and grid based clustering algorithm finds clusters automatically. By setting a density threshold as noise, the proposed method is also capable of detecting white and non-white noise. Without the calculation of Euclidean mutual distances between patterns, it successfully decreases enormously computation complexity to the equation $O(n * nznode)$. Besides, a soft decision strategy using fuzzy approximation is proposed to compute node's density instead of simply counting the number of data in the cell or computing all other data density contributions. It turns out that the presented density and grid based clustering method is capable of clustering datasets with arbitrary shapes, outperforming the conventional K-means, FSFDP and DBSCAN methods.

- Improved clustering accuracy.

As it is not an easy job to classify data in just one single run, a two-stage clustering strategy is developed based on density and the minimum internal and external distance ratio. In the first-stage, density-based partitional clustering can provide reasonable/satisfying partitional clustering results and even perfect results as desired. A new concept of minimum internal and external distance ratio, called cross-distance-ratio, is used to determine to merge two partitioned clusters or not. So in the second stage, it will determine whether to merge some partitioned

clusters or not based on the cross-distance-ratio curves. Besides, an effective way to identify the separation inside a larger cluster and between clusters is described as well by analyzing the external density and internal density. Based on two facts that the densities are smoothly changing inside one cluster and the distances, as well as the densities, are significantly different between clusters, the proposed clustering algorithm is capable of generating satisfying results. Therefore, the presented clustering algorithm based on density and the minimum internal and external distance ratio can be considered as a replacement of traditional complex shape clustering algorithms.

- Found a more accurate solution for MDS purpose.
- Enlarged the cluster separation regions in the embedding results.

High dimensional data cannot be viewed on a two-dimensional plane. In order to see these data, dimension reduction is needed. Multi-Dimensional Scaling (MDS) techniques are explored to visualize high dimensional data in a low dimensional space. In this work, density concentration is incorporated into the improved MDS with LM optimization. Different from other traditional MDS approaches, the presented algorithm involves an unsupervised approach along the process of MDS, without extra computations. It reveals both the distance relationships as well as the density relationships between data categories, and therefore successfully provide us a better visualization of data. Compared with other state-of-the-art MDS methods, our proposed algorithm achieves the best performance. More importantly, it provides an integrated density based MDS approach to performing small cluster formations and embedding simultaneously without any prior knowledge, which can be easily applied to project data into any desired space. Besides, the proposed algorithm can be very useful and heuristic for combining many other unsupervised clustering methods with traditional MDS techniques in further study.

- Reveal data relations as a graph in the embedding results.

Moreover, nearest neighbor maps are generated along the presented MDS algorithm, which can provide data nearest neighbor information in density field, showing how data are connected to the others. Unlike the conventional MDS approaches with only one purpose of embedding data, the proposed MDS algorithm is capable of showing how the data are related/connected to the others. With the connections between related data shown, more information can be revealed in the embedded results.

7.2 Suggestion for Future Work

More discussion on the determination of grid size is needed in the presented grid-based clustering approaches. In this work, settings on grid size based on empirical analysis are demonstrated other than numerically analyzed grid size.

Partitional clustering approaches often provide us high clustering accuracy but they also suffer from heavy computation cost. Therefore, accelerated implementation of calculating the minimum internal and external distance ratios is highly expected.

Develop fast high-dimensional data clustering strategy for image recognition and many other problems. It's critical to investigate the relevance of high dimensions for clustering purpose.

Explore other strategies to fuse clustering methodology into visualization techniques. There are different options for fusing clustering into visualization other than linear combination or separate optimization.

It is worth embedding clustering/visualization algorithms into artificial neural networks for many purposes, e.g. classification, dimension reduction, prediction etc.

REFERENCES

[1] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press. pp. 281-297, 1967.

[2] Stephen C. Johnson, "Hierarchical Clustering Schemes," Psychometrika, pp. 241-254, 32(3), Sep., 1967.

[3] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," Journal of the Royal Statistical Society, Series B 39 (1): 1-38, 1977.

[4] J. C. Dunn, “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters,” Journal of Cybernetics, 3 (3): 32-57, 1973.

[5] Wang, Yunhe, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. "Cnnpack: Packing convolutional neural networks in the frequency domain," In Advances in neural information processing systems, pp. 253-261. 2016.

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks," In Advances in neural information processing systems, pp. 1097-1105. 2012.

[7] Jiang, Zhuxi, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. "Variational deep embedding: An unsupervised and generative approach to clustering." arXiv preprint arXiv:1611.05148, 2016.

[8] Xie, Junyuan, Ross Girshick, and Ali Farhadi. "Unsupervised deep embedding for clustering analysis." In International conference on machine learning, pp. 478-487. 2016.

[9] Kruskal, Joseph B. "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis." Psychometrika 29, no. 1, pp. 1-27, 1964.

[10] J.W. Sammon Jr., "A nonlinear mapping for data structure analysis," IEEE Transactions on Computers 18(5): 401-409, 1969.

[11] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, vol. 290 , pp. 2319-2323, 2000

[12] David L. Donoho and Carrie Grimes, "Hessian eigenmaps: Locally linear embeding techniques for high-dimensional data," Proc Natl Acad Sci, vol. 100, no. 10, pp. 5591–5596, 2003.

[13] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, pp. 226-231, 1996.

[14] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492-1496, 2014

[15] W. Zhang, and J. Li, "Extended fast search clustering algorithm: widely density clusters, no density peaks," Computer Science & Information Technology (CS & IT), arXiv:1505.05610, 2015, pp. 01–17, doi. 10.5121/csit.2015.50701, 2015

[16] Jiang, Zhuxi, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. "Variational deep embedding: An unsupervised and generative approach to clustering." arXiv preprint arXiv:1611.05148, 2016.

[17] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological): 1-38, 1977.

[18] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) pp. 1-38, 1977.

[19] Van Thoai, Nguyen, and Hoang Tuy. "Convergent algorithms for minimizing a concave function." Mathematics of operations Research 5, no. 4, pp. 556-566, 1980.

[20] Kuczma, Marek. An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality. Springer Science & Business Media, 2009.

[21] Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation." IEEE Transactions on pattern analysis and machine intelligence, vol. 22, no. 8: 888-905, 2000.

[22] Dunn, Joseph C. "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters." pp. 32-57, 1973.

[23] Agrawal, Rakesh, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. Vol. 27, no. 2. ACM, 1998.

[24] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." In Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487-499. 1994.

[25] I.T. Jolliffe, “Principal Component Analysis,” Springer-Verlag, pp. 487, 1986.

[26] Daniel D. Lee and H. Sebastian Seung, “Algorithms for Non-negative Matrix Factorization,” Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference, pp. 556-562, 2001.

[27] Fisher, Ronald A. "The use of multiple measurements in taxonomic problems." Annals of human genetics 7, no. 2 (1936): 179-188.

[28] Hinton, Geoffrey E., and Sam T. Roweis. "Stochastic neighbor embedding." In Advances in neural information processing systems, pp. 857-864. 2003.

[29] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research, no.9, pp. 2579-2605, Nov. 2008.

[30] Kruskal, Joseph B., and Myron Wish. Multidimensional scaling. Vol. 11. Sage, 1978.

[31] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Numerische mathematik, vol.1, no. 1,pp. 269-271, 1959.

[32] Floyd, Robert W. "Algorithm 97: shortest path." Communications of the ACM, vol. 5, no. 6, pp. 345, 1962.

[33] P. Joia, F. Paulovich, D. Coimbra, J.A. Cuminato, and L.G. Nonato, “Local Affine Multidimensional Projection,” IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2563-2571, Dec. 2011.

[34] Gower, John C., and Garmt B. Dijksterhuis. Procrustes problems. Vol. 30. Oxford University Press on Demand, 2004..

[35] Golub, Gene H., and Christian Reinsch. "Singular value decomposition and least squares solutions." Numerische mathematik, vol. 14, no. 5, pp. 403-420, 1970.

[36] E. Schikuta and M. Erhart, "The Bang-Clustering System: Grid-Based Data Analysis," Springer Verlag, 1997.

[37] Jané, Raimon, Pablo Laguna, Nitish V. Thakor, and Pere Caminal. "Adaptive baseline wander removal in the ECG: Comparative analysis with cubic spline technique." In Computers in Cardiology 1992, Proceedings of, pp. 143-146. IEEE, 1992.

[38] Kevin J. Lang and Michael J, Witbrock, "Learning to Tell Two Spirals Apart," in Proceedings of the 1988 Connectionist Models Summer School, 1988.

[39] Limin Fu and Enzo Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," BMC Bioinform 8(1): 3, 2007.

[40] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," Pattern Recognition, 39(5), pp. 761-775, 2006.

[41] A. Gionis, H. Mannila, P. Tsaparas, "Clustering aggregation," ACM Trans. Knowl. Discovery Data, 1:109–117, 2007.

[42] Huijun Gao, Changxing Ding, Chunwei Song, and Jiangyuan Mei, "Automated Inspection of E-Shaped Magnetic Core Elements Using K-tSL-Center Clustering and Active Shape Models," IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1782-1789, Aug. 2013

[43] Dumidu Wijayasekara, Ondrej Linda, Milos Manic, and Craig Rieger, “Mining Building Energy Management System Data Using Fuzzy Anomaly Detection and Linguistic Descriptions,” IEEE Transactions on Industrial Informatics, vol. 10, no. 3, pp. 1829-1840, Aug. 2014.

[44] Jinlin Zhu, Zhiqiang Ge and Zhihuan Song, “HMM-Driven Robust Probabilistic Principal Component Analyzer for Dynamic Process Fault Classification,” IEEE Transactions on Industrial Electronics, vol. 62, no. 6, pp. 3814-3821, June 2015.

[45] Kangkang Zhang, R. Gonzalez, Biao Huang and Guoli Ji, “Expectation-Maximization Approach to Fault Diagnosis With Missing Data,” IEEE Transactions on Industrial Electronics, vol. 62, no. 2, pp. 1231-1240, Feb. 2015.

[46] Chun-Cheng Lin, Der-Jiunn Deng, Jia-Rong Kang, Sen-Chia Chang, and Chuang-Hua Chueh, “Forecasting Rare Faults of Critical Components in LED Epitaxy Plants Using a Hybrid Grey Forecasting and Harmony Search Approach,” IEEE Transactions on Industrial Informatics, vol. PP , issue. 99, pp. 1-8, Dec. 2015.

[47] Wenjie Bi, Meili Cai, Mengqi Liu, and Guo Li, “A Big Data Clustering Algorithm for Mitigating the Risk of Customer Churn,” IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1270-1281, June 2016.

[48] E. Schikuta, “Grid clustering: An efficient hierarchical clustering method for very large data sets,” in Proc. 13th Int.Conf. on Pattern Recognition, vol. 2, pp. 101-105, 1996.

[49] W. Wang, J. Yang, and R. Muntz, “STING: A Ststistical Information Grid Approach to Spatial Data Mining,” In Proceedings of 23rd VLDB Conference, pp. 186-195, Athens, Greece, 1997.

[50] D.C Hernandez, Van-Dung Hoang, A. Filonenko, and Kang-Hyun Jo, "Vision-based heading angle estimation for an autonomous mobile robots navigation," ISIE 2014, pp. 1967-1972, 1-4 June, 2014.

[51] Online material at https://www.mountain-forecast.com/mountainobjects/Big-Savage-Mountain.jpg, last viewed on Feb. 12, 2018.

[52] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," Proceedings of the 1990 ACM SIGMOD international conference on Management of data, pp.322-331, May 23-26, 1990.

[53] G. Karypis, E.-H. Han and V. Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling," Computer, vol. 32, no. 8, pp. 68-75, Aug. 1999.

[54] R. Ibrahim, N. Ahmed, N.A. Yousri, M.A. Ismail, "Incremental mitosis: discovering clusters of arbitrary shapes and densities in dynamic data," 11th International Conference on Machine Learning and Applications, vol. 1, pp. 102-107, 2012.

[55] S. Ayramo and T. Karkkainen, "Introduction to partitioning-based cluster analysis methods with a robust example," Reports of the Department of Mathematical Information Technology; Series C: Software and Computational Engineering, C1, 1-36, 2006.

[56] Yoshihiro Nakamura and Osamu Hasegawa, "Nonparametric Density Estimation Based on Self-Organizing Incremental Neural Network for Large Noisy Data," IEEE Trans on Neural Networks and Learning Systems, pp. 8-17, vol. 28, no. 1, Jan., 2017.

[57] Siamak Mehrkanoon, Carlos Alzate, Raghvendra Mall, Rocco Langone, and Johan A. K. Suykens, "Multiclass Semisupervised Learning Based Upon Kernel Spectral Clustering," IEEE Trans on Neural Networks and Learning Systems, pp. 720-732, vol. 26 no. 4, April 2015

[58] Zhengming Li, Zhihui Lai, Yong Xu, Jian Yang and David Zhang, "A Locality-Constrained and Label Embedding Dictionary Learning Algorithm for Image Classification," IEEE Trans on Neural Networks and Learning Systems, pp. 278-293, vol. 28, no. 2, Feb., 2017.

[59] Xiaozhao Fang, Shaohua Teng, Zhihui Lai, Zhaoshui He, Shengli Xie and Wai Keung Wong, "Robust Latent Subspace Learning for Image Classification," IEEE Trans on Neural Networks and Learning Systems, DOI: 10.1109/TNNLS.2017.2693221, May, 2017.

[60] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in Processings of 1994 IEEE Workshop on Applications of Computer Vision, pp. 138-142, 1994.

[61] Kuhn, Harold W. The hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2):83-97, 1955.

[62] Jain, Anil K., and Martin HC Law, "Data clustering: A user's dilemma," In International conference on pattern recognition and machine intelligence, pp. 1-10. Springer, Berlin, Heidelberg, 2005.

[63] Moutarde, Fabien, and Alfred Ultsch, "U*F clustering: a new performant cluster-mining method based on segmentation of Self-Organizing Maps," In Proceedings of the 5th Workshop On Self-Organizing Maps (WSOM'05), pp. 25-32, Paris, France, 2005.

[64] Dominik Ja¨ckle, Fabian Fischer, Tobias Schreck, and Daniel A. Keim, “Temporal MDS Plots for Analysis of Multivariate Data,” IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 1, pp. 141-150, Jan. 2016.

[65] Chris Bryan, Kwan-Liu Ma, and Jonathan Woodring, “Temporal Summary Images: An Approach to Narrative Visualization via Interactive Annotation Generation and Placement,” IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 511-520, Jan. 2017.

[66] Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park, “UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization,” IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 1992-2001, Dec. 2013.

[67] Tim Gerrits, Christian Ro¨ssl, and Holger Theisel, “Glyphs for General Second-Order 2D and 3D Tensors,” IEEE Transactions on Visualization and Computer Graphics,vol. 23, no. 1, pp. 980-989, Jan. 2017.

[68] D. Marquardt, “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” Journal of the Society for Industrial and Applied Mathematics, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[69] B. M. Wilamowski and H. Yu, “Improved Computation for Levenberg Marquardt Training,” IEEE Transactions on Neural Networks, vol. 21, no. 6, pp. 930–937, Jun. 2010.

[70] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. “A Public Domain Dataset for Human Activity Recognition Using Smartphones,” 21th European

Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 24-26, April 2013.

[71] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges, online at http://yann.lecun.com/exdb/mnist/, THE MNIST DATABASE of handwritten digits, last seen on 16 January 2018.

[72] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in Processings of 1994 IEEE Workshop on Applications of Computer Vision, pp. 138-142, 1994.

[73] Online resource at https://corpus.byu.edu/wiki/. Last seen on Feb. 4, 2018.