

ขั้นตอนวิธีการเรียนรู้แบบไม่มีผู้สอนสำหรับการเกาะกลุ่มข้อมูลอย่างทนทานและ
การประมาณจำนวนกลุ่มที่เป็นไปได้



นางสาวอุรวิรัฐ วัฒนชนม์

สถาบันวิทยบริการ

จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์


คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2549

ISBN 974-14-3425-1

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

AN UNSUPERVISED LEARNING ALGORITHM FOR ROBUST CLUSTERING AND
ESTIMATING THE FEASIBLE NUMBER OF CLUSTERS



Ms. Ureerat Wattanachon

สภามหาวิทยาลัย
จุฬาลงกรณ์มหาวิทยาลัย

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science
Department of Mathematics

Faculty of Science

Chulalongkorn University

Academic year 2006

ISBN 974-14-3425-1

Copyright of Chulalongkorn University

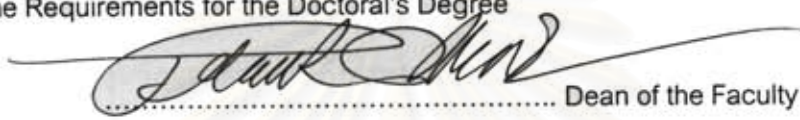
Thesis Title AN UNSUPERVISED LEARNING ALGORITHM FOR ROBUST CLUSTERING AND ESTIMATING THE FEASIBLE NUMBER OF CLUSTERS

By Ms. Ureerat Wattanachon

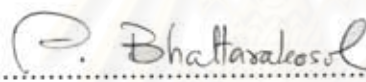
Field of Study Computer Science

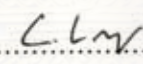
Thesis Advisor Professor Chidchanok Lursinsap, Ph.D.

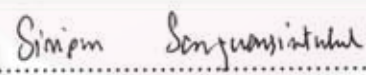
Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral's Degree



..... Dean of the Faculty of Science
(Professor Piamsak Menasveta, Ph.D.)


THESIS COMMITTEE

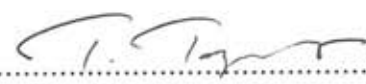

..... Chairman
(Assistant Professor Pattarasinee Bhattarakosol, Ph.D.)


..... Thesis Advisor
(Professor Chidchanok Lursinsap, Ph.D.)


..... Member
(Siripun Sanguansintukul, Ph.D.)


..... Member
(Associate Professor Boonserm Kijsirikul, Ph.D.)


..... Member
(Associate Professor Kosin Chamnongthai, Ph.D.)


..... Member
(Assistant Professor Thitipong Tanprasert, Ph.D.)

อุไรรัฐ วัฒนชนม์ : ขั้นตอนวิธีการเรียนรู้แบบไม่มีผู้สอนสำหรับการเกาะกลุ่มข้อมูลอย่าง
ทนทานและการประมาณจำนวนกลุ่มที่เป็นไปได้ (AN UNSUPERVISED LEARNING
ALGORITHM FOR ROBUST CLUSTERING AND ESTIMATING THE FEASIBLE
NUMBER OF CLUSTERS). อ. ที่ปรึกษา: ศ. ดร. ชิดชนก เหลือสินทรัพย์, 99 หน้า.
ISBN 974-14-3425-1.

การเกาะกลุ่มข้อมูลเป็นวิธีการที่ใช้ในการแบ่งกลุ่มเซตของข้อมูล โดยที่ข้อมูลที่อยู่ในกลุ่ม
เดียวกันจะมีลักษณะที่คล้ายคลึงกันและข้อมูลที่อยู่ต่างกลุ่มกันจะมีลักษณะที่แตกต่างกันที่สุดที่เป็นไป
ได้ ซึ่งขั้นตอนวิธีการเกาะกลุ่มข้อมูลที่มีอยู่ อาทิเช่น single-link, k-mean, CURE และ CSM ได้
ออกแบบให้มีการแบ่งกลุ่มข้อมูลโดยขึ้นอยู่กับค่าพารามิเตอร์ที่ต้องกำหนดโดยผู้ใช้ ขั้นตอนวิธีการ
เหล่านี้ไม่สามารถแบ่งกลุ่มข้อมูลได้ถูกต้องถ้าค่าพารามิเตอร์ที่ต้องกำหนดไม่เหมาะสมกับลักษณะของ
ข้อมูลที่จะทำการแบ่งกลุ่ม ขั้นตอนวิธีเหล่านี้ส่วนใหญ่สามารถแบ่งกลุ่มได้ดีกับข้อมูลที่มีลักษณะการ
เกาะตัวภายในกลุ่มที่แน่นและเป็นทรงกลม ดังนั้นวิทยานิพนธ์นี้จึงเป็นการนำเสนอขั้นตอนวิธีการเกาะ
กลุ่มแบบผสมผสานแบบใหม่ที่เรียกว่า "Self-Partition and Self-Merging" หรือ เอสทีเอสเอ็ม
(SPSM) ซึ่งในเฟสแรก ขั้นตอนวิธีเอสทีเอสเอ็มจะทำการแบ่งกลุ่มเซตของข้อมูลออกเป็นกลุ่มย่อยเล็ก
ๆ หลายกลุ่ม จากนั้นในเฟสสองจะทำการตัดข้อมูลและกลุ่มย่อยเล็ก ๆ ที่เป็นสัญญาณรบกวนออกไป
ส่วนในเฟสสามกลุ่มย่อยเล็ก ๆ ที่มีลักษณะการเกาะตัวของข้อมูลที่แน่น จะทำการรวมกลุ่มเหล่านี้เข้า
ด้วยกันเรื่อย ๆ ให้ได้กลุ่มที่ใหญ่ขึ้น โดยการรวมกลุ่มจะพิจารณาจากระยะห่างระหว่างกลุ่มและ
ระยะห่างภายในกลุ่ม จากผลการทดลองพบว่า ขั้นตอนวิธีเอสทีเอสเอ็ม มีประสิทธิภาพในการจัดการ
กับข้อมูลที่มีสัญญาณรบกวน นอกจากนี้ ขั้นตอนวิธีเอสทีเอสเอ็มยังสามารถแบ่งกลุ่มเซตของข้อมูลที่มี
ลักษณะรูปร่างและความหนาแน่นของข้อมูลที่แตกต่างกัน รวมทั้งยังคงทนต่อข้อมูลที่มีสัญญาณ
รบกวน และให้ผลการเกาะกลุ่มที่ดีกว่าการเกาะกลุ่มจากขั้นตอนวิธีการเกาะกลุ่มอื่น ๆ ส่วนความ
ซับซ้อนของเวลาที่ใช้ของขั้นตอนวิธีเอสทีเอสเอ็ม คือ $O(N^2)$ เมื่อ N แทนจำนวนข้อมูลทั้งหมด

ภาควิชา.....คณิตศาสตร์.....ลายมือชื่อนิสิต.....อุไรรัฐ วัฒนชนม์.....
สาขาวิชา.....วิทยาการคอมพิวเตอร์.....ลายมือชื่ออาจารย์ที่ปรึกษา.....C.L.W.....
ปีการศึกษา 2549

4473861523 : MAJOR COMPUTER SCIENCE

KEY WORD: UNSUPERVISED LEARNING / CLUSTERING.

UREERAT WATTANACHON: AN UNSUPERVISED LEARNING ALGORITHM FOR ROBUST CLUSTERING AND ESTIMATING THE FEASIBLE NUMBER OF CLUSTERS. THESIS ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., 99 pp. ISBN 974-14-3425-1.

Data clustering is a discovery process that groups a set of data such that the data points in the same cluster are as similar as possible and the data points of different clusters are as dissimilar as possible. Existing clustering algorithms, such as single-link clustering, k-means, CURE, and CSM are designed to find clusters based on pre-defined parameters specified by users. These algorithms can breakdown if the choice of parameters is incorrect with respect to the data set being clustered. Most of these algorithms work very well for compact and hyperspherical clusters. In this dissertation, the new hybrid clustering algorithm called "Self-Partition and Self-Merging" (SPSM) is proposed. The SPSM algorithm partitions the input data set into several subclusters in the first phase, and then removes the noisy data and the noisy subclusters in the second phase. In the third phase, the dense subclusters are continuously merged to form the larger clusters based on the inter-distance and intra-distance criteria. From the experimental results, the SPSM algorithm is very efficient to handle the noisy data set. Moreover, the SPSM algorithm is able to cluster the data sets of arbitrary shapes and different density very efficiently, tolerate to noise, and provide better clustering results than the existing clustering algorithms. The computational complexity of the SPSM algorithm is $O(N^2)$, where N is the number of data points.

Department :Mathematics..... Student's Signature :*อุรีรัตน์ วัฒนอักษร*.....

Field of Study :Computer Science.....Advisor's Signature :*C.L.W*.....

Academic Year : 2006

Acknowledgements

During my years as a Ph.D. student, I have received a lot of tuition, care and friendship from several people, some of which I wish to thank here.

- First of all I would like to thank Development and Promotion for Science and Technology talents project (DPST) of Thailand who sponsor the research scholarships.
- During my time as a Ph.D.s student, I am grateful to my supervisor, Prof.Dr. Chidchanok Lursinsap, to whom with his advice, guidance and care, help me to overcome the necessary difficulties of the process of research and make this dissertation possible.
- I would like to thank Dr.Mikael Boden at University of Queensland, Australia, who gives me a wonderful suggestions in Ph.D. research methodologies.
- My thanks also goes to dissertation committee with their advice and guidance, help focus my research activities.
- I would also like to thank Supaporn Bunrit, Benjamas Panyangam and all my colleagues at the Advanced Virtual Intelligent Computing (AVIC) Center, Department of Mathematics, Chulalongkorn University, who give me a number of useful suggestions. Special thanks also goes to Sasithorn Anantasopon, my room-mate, for her wormest care support.
- Finally, my deepest gratitude goes to Wattanachon's family, for their sponsor, love and care and especially Mr.Jakkarin Suksawatchon, for his love, wormest care support and being patient during my doubtful stage.

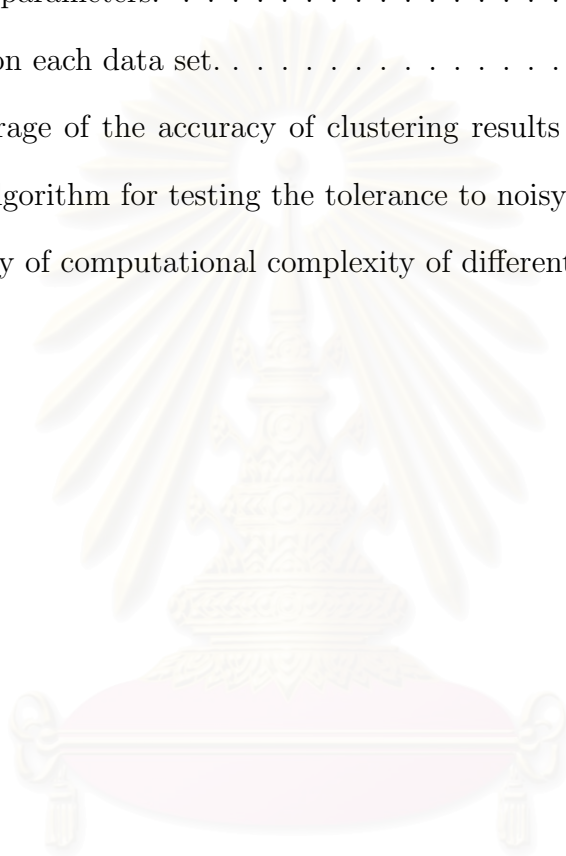
Table of Contents

Acknowledgements	vi
Table of Contents	vii
List of Tables	ix
List of Figures	x
1 INTRODUCTION	1
1.1 Introduction and Problem Review	1
1.2 Research Objective	5
1.3 Scopes of the Study	5
1.4 Research Plans	6
1.5 Research Advantages	6
2 LITERATURE REVIEWS	7
2.1 Hierarchical Clustering Algorithms	7
2.2 Partitional Clustering Algorithms	9
2.3 Hybrid Clustering Algorithms	10
2.4 Self-Organizing Map	12
2.5 Variants of SOM	15
3 PROPOSED METHOD	21
3.1 Phase 1: Self-Partition	24
3.1.1 Architecture of the DTS-SOM	24
3.1.2 Training of the DTS-SOM	25
3.1.3 Data Decomposition	30
3.2 Phase 2: Noise Removal	30
3.2.1 Density Computation	32
3.2.2 Cluster Separation	34

3.2.3	Cluster Verification	38
3.3	Phase 3: Self-Merging	39
3.3.1	Neighboring Merging	40
3.3.2	Local Merging I	46
3.3.3	Local Merging II	48
3.3.4	Refinement Merging	52
3.4	Clustering Example	54
4	EXPERIMENTAL RESULTS	59
4.1	Experiment 1	60
4.2	Experiment 2	60
4.3	Experiment 3	61
4.4	Experiment 4	66
4.5	Experiment 5	69
4.6	Experiment 6	70
4.7	Experiment 7	71
4.8	Complexity Analysis	74
5	CONCLUSION	78
	References	81
	Biography	85

List of Tables

2.1	Types of definition of inter-cluster distance.	9
4.1	Phase 1 parameters.	60
4.2	Details on each data set.	61
4.3	The average of the accuracy of clustering results obtained through the SPSM algorithm for testing the tolerance to noisy data.	73
4.4	Summary of computational complexity of different algorithms.	76



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

List of Figures

2.1	The hierarchical clustering algorithm for a data set of seven points. (a) The input data set. (b) A possible dendrogram.	8
2.2	The most common topologies.	13
2.3	Topological map with 10 neurons at certain learning stage. A is the best matching unit, and B, C, D, E, F , and G are its direct neighbors.	16
2.4	Weight initialization in the CSG algorithm for the two-dimensional input data. (a) Neuron A is to be split to generate four new neurons on the topological map. (b) The best matching unit A and the new neurons (1, 2, 3, and 4) in the two-dimensional input space before and after splitting. ML, MR, MT , and MB are middle-points between neuron A and its direct left, right, top and bottom neurons, respectively. Dashed lines ($B1, 1D, D2, 2C, C4, 4E, E3$, and $3B$) denote the neuron connections after splitting. (c) New neurons (1, 2, 3, and 4) with new connections after splitting on the output map.	17
2.5	Learning by CSG algorithm. (a) Output map. (b) Input space with neuron connections.	18
2.6	Fundamental operations of the ETree. (a) Best matching unit search. (b) Tree distance.	20
3.1	The overview of the proposed method – SPSM.	21
3.2	An example of a sparse noisy subcluster generated in Phase 1.	23
3.3	The overall steps of Phase 1.	24

3.4	Tree search operation of the DTS-SOM: how the best matching unit is found. White and black nodes are denoted the internal nodes and the leaf nodes, respectively. The arrows show tree search path to find the best matching unit.	26
3.5	An example of the DTS-SOM learning. (a) The tree structure. (b) Input space with the leaf neuron connections. A number shows the neuron index.	28
3.6	The overall steps of Phase 2.	32
3.7	An example of a point x_i is outside the hyperbox.	33
3.8	An example of the density arrangement. (a) Four types of sorted density values. (b) The elliptic region denoted by dashed line shows the expected area which may be the noisy subclusters.	36
3.9	The overall steps of Phase 3.	41
3.10	An example of the partial regions (connected regions). (a) The partial region of non-overlapping subclusters. (b) The partial region of overlapping subclusters.	42
3.11	An example of the ranges of each component for two-dimensional data space. (a) The minimum and maximum positions of the first component. (b) The minimum and maximum positions of the second component. . .	43
3.12	An example of the partial region (connected region). The gray dots in the connected region are used for computing the inter-distance.	45
3.13	An example how the SPSM works. (a) Input data set of three clusters. (b) The lateral connections among the leaf nodes. A number shows the node index. (c) The set of 100 subclusters obtained through Phase 1. (d) The set of 50 dense subclusters. Each subcluster is denoted by a color. Each star is represented a prototype vector obtained from the DTS-SOM Training process and each square is represented a subcluster center.	56

4.1	Clustering result on the experiment 1. (a) Data Set 1. (b) SPSM on Data Set 1 (3 clusters). Each cluster is denoted by a symbol.	62
4.2	Clustering results on the experiment 2 for Data Set 2 and Data Set 3. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.	63
4.3	Clustering results on the experiment 2 for Data Set 4. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.	64
4.4	Clustering results on the experiment 3 produced in SPSM algorithm. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.	65
4.5	Clustering results on the experiment 3 produced in SPSM algorithm. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.	66
4.6	Clustering results on the experiment 3. (a) and (b) are the final decomposition of Data Set 6 and Data Set 8 obtained from the single-link algorithm, respectively. (c) and (d) are the clustering results of Data Set 7 and Data Set 8 acquired from the complete-link algorithm, respectively. Each cluster is denoted by a symbol.	67
4.7	Clustering results on the experiment 3 obtained through the algorithm CURE. (a) and (b) are the final decomposition of Data Set 6 and Data Set 7, respectively. Each cluster is denoted by a symbol.	68
4.8	Clustering results on the experiment 3 produced by the algorithm CSM. (a) and (b) are the final decomposition of Data Set 7 and Data Set 8, respectively. Each cluster is denoted by a symbol.	68
4.9	Clustering result on the experiment 4 for Data Set 5. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.	69

- 4.10 Clustering result on the experiment 5 for Data Set 9. Each cluster is denoted by a symbol. The star symbol is represented noisy data points. 70
- 4.11 Clustering results on the experiment 6 obtained through the algorithm SPSM. (a) and (b) are the final decompositions of Data Set 7 with different stopping criteria. Each cluster is denoted by a symbol. 71
- 4.12 Clustering results on the experiment 6 obtained through the algorithm SPSM. (a) and (b) are the final decompositions of Data Set 6 and Data Set 8 with different initialization of the first neuron. Each cluster is denoted by a symbol. 72
- 4.13 An example of data set used in the experiment 7. The dots represented the data points and the stars denoted the noisy data points 73

CHAPTER I

INTRODUCTION

1.1 Introduction and Problem Review

Hugh amounts of data collected and stored in databases will increase the need for efficient and effective analysis methods to make use of the information implicitly contained in the data for further analysis and management. One of the primary data analysis tasks is cluster analysis which is intended to help users to understand the natural grouping or structure in a data set [1, 2]. Therefore, the development of improved clustering algorithm has received a lot of attentions in the last few years. The goal of clustering algorithm is to partition data into groups or clusters such that the data points in the same cluster are as similar as possible and the data points of different clusters are as dissimilar as possible. In the clustering process, there are no pre-defined classes and no examples, also known as unsupervised classification, that would show what kind of desirable relations should be valid among the data [3, 4]. The applications of clustering algorithm can be found in areas such as grouping, decision-making, and machine learning, including data mining, document retrieval, image segmentation, and pattern classification [1, 3, 5, 6, 7].

Many data clustering algorithms have been proposed in the literatures. These algorithms can be categorized into partitional clustering [8, 9], hierarchical clustering [10, 11, 12], artificial neural networks for clustering [13, 14, 15], statistical clustering algorithms [16, 17], fuzzy clustering [18], density-based clustering algorithms [19, 20, 21], and so on. In these methods, hierarchical and partitional clustering algorithms are two

primary approaches in research communities.

Hierarchical clustering treats each data point as a singleton cluster, and then successively merges clusters until all points have been merged into a single remaining cluster. So, hierarchical clustering algorithms organize data into a hierarchical structure according to the proximity matrix. The results of hierarchical clustering are usually depicted by a binary tree or *dendrogram*. The root node of the dendrogram represents the whole data set and each leaf node is regarded as a data point. The intermediate nodes describe the extent that the data are proximal to each other; and the height of the dendrogram usually expresses the distance between each pair of data points or clusters, or an data point and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels. Therefore, cutting the tree at particular level produces a partition into g disjoint groups. With hierarchical structure, the different clustering results can be obtained for different similarity requirements. The well-known hierarchical clustering algorithms are single-link, complete-link, average-link, and so on. Besides, if the clusters are close to one another even by noises, or if their shapes and sizes are not hyperspherical, the hierarchical clustering algorithms give uncorrectness results. However, most of the hierarchical clustering algorithms require time complexity of $O(N^2 \log N)$, where N is the number of input data points.

On the other hand, partitional clustering attempts to break a data set into k clusters such that the partition optimizes a given criterion [22]. The *k-means* algorithm is one of the most famous partitional clustering algorithms [23]. Because the *k-means* algorithm is very simple and can be easily implemented. Although widely used, the *k-means* algorithm suffers from some drawbacks [7]. There is no efficient method for identifying the initial partitions and the number of clusters k . It can work very well for compact and hyperspherical clusters. Moreover, *k-means* is sensitive to noises. However, most partitional algorithms have advantage on the execution time which is in linear time.

Several clustering methods have been proposed to combine the features of hierarchical and partitional clustering algorithms. It called the *hybrid* idea or two-phase clustering algorithm. In general, these algorithms first partition the input data set into pre-defined small subclusters instead of using all the data points as the distinct clusters. Then, these algorithms construct a hierarchical structure based on these small subclusters and these subclusters are next grouped into larger clusters.

Algorithm *Clustering Using Representatives* (CURE) is one of the famous two-phase clustering algorithm [10]. In CURE, instead of using all data points to represent the clusters, a constant number of well scattered points are chosen to represent a cluster. Then the chosen scattered points are shrunk towards the center of the cluster by a shrinking factor in order to eliminate the effects of noise. These scattered points after shrinking are used as representatives of the cluster. The clusters with the closest pair of representative points are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. A major limitation of the CURE algorithm is that the merging decisions are based upon input parameters which are a shrinking factor, the number of representative points, the number of clusters. Thus, these parameters have to carefully choose.

Another hybrid clustering algorithm has been proposed to remedy the drawbacks of hierarchical and partitional clustering while combining their advantages [24]. The new similarity measure between two clusters is proposed, namely, *cohesion* to measure the inter-cluster distance. Using cohesion, a two-phase clustering algorithm is presented called *Cohesion-based Self-Merging* (CSM). Algorithm CSM starts with partitioning the input data set into several small subclusters in the first phase and then continuously merges the subclusters based on cohesion in a hierarchical manner in the second phase. During merging process, algorithm CSM removes all the subclusters whose sizes are less than the threshold value. Therefore, algorithm CSM is able to deal with noisy data

set. Like CURE, algorithm CSM suffers from parameter settings. Note that the number of subclusters, the desired number of clusters, the impedance factor, and the threshold value are parameters specified by users. With proper parameters, algorithm CSM can give a good results. In practice, it is not easy to choose the suitable parameters.

However, both CURE and CSM algorithms do not produce a suitable estimation of the number of output clusters by themselves and it has to be provided as an input parameter. Moreover, these algorithms fail if the choice of parameters is incorrect with respect to the data set being clustered. To avoid the limitation of parameter settings and to effectively alleviate the disadvantages of hierarchical and partitional clustering algorithms, the new hybrid clustering called “*Self-Partition and Self-Merging*” (abbreviated as SPSM) is proposed. The SPSM algorithm has been designed into three phases. In Phase 1, the new partitional clustering is introduced based on a self-creating and self-organizing algorithm designed to improve SOM algorithm called *Dynamic Tree-Structured Self-Organizing Map* (DTS-SOM). Once the DTS-SOM performed, the number of initial subclusters is automatically obtained. To achieve a better clustering result and be less affected by noises, the noisy data and the noisy subclusters are removed by Phase 2. Then, algorithm SPSM performs self-merging process in Phase 3 based on inter-distance and intra-distance criteria. The SPSM algorithm automatically obtains the final clusters and can identify the noisy data. The main contributions of our proposed method can be summarized as follows:

- The DTS-SOM is proposed to cope with the initialization of the number of clusters required in the partitional clustering algorithm. The DTS-SOM is a variant of SOM which is a self-creating and self-organizing algorithm designed to improve the SOM algorithm. Using DTS-SOM, it is able to overcome the limitations of SOM, because the SOM must pre-define the topology structure and the number of neurons before the training process.

- The SPSM algorithm also proposes the noise removal method which can deal with the noisy data set. So algorithm SPSM is able to not only resist noises, but also lead to good clustering results.
- The SPSM algorithm is able to cluster the data sets of arbitrary shapes very efficient and provide better results than the other algorithms.
- The parameter settings of our proposed method are minimum. The parameter requirements are default used for the DTS-SOM training.

1.2 Research Objective

The objective of this dissertation prospectus is to develop a new unsupervised learning algorithm that can be robust in three aspects:

- (a) Robust to the initialization (the number of clusters),
- (b) Robust to cluster shapes (ability to detect arbitrary shapes of clusters), and
- (c) Robust to noisy points (ability to tolerate noise).

1.3 Scopes of the Study

In this dissertation, the scope of work is constrained as follows:

1. This proposed algorithm is an unsupervised learning algorithm.
2. The number of dimensions and training data are finite.
3. The performance results from the proposed algorithm are compared with the results by existing agglomerative clustering techniques.

1.4 Research Plans

1. Study related papers and documents to unsupervised learning algorithms.
2. Develop a new unsupervised learning algorithm.
3. Experiment with benchmark data and compare the results with those from the other algorithms.
4. Analyze the experimental results and conclude the outcomes.

1.5 Research Advantages

It is expected that the new approach and prototype are

1. A new unsupervised method is able to deal with unbalanced and irregular clusters.
2. The number of clusters is automatically derived by the proposed algorithm.
3. This algorithm can be used to solve in clustering problems.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER II

LITERATURE REVIEWS

In this chapter, the existing clustering algorithms (hierarchical clustering algorithms, partitional clustering algorithms, and hybrid clustering algorithms) including Self-Organizing Map (SOM) and its variants, are briefly revised.

2.1 Hierarchical Clustering Algorithms

Hierarchical clustering begins with each input data point in a distinct (singleton) cluster, and then successively merges clusters together until a stopping criterion is satisfied that is all points have been merged into a single remaining cluster. A hierarchical clustering is often represented as a binary tree or dendrogram [1]. The root node of the dendrogram represents the whole data set and each leaf node is regarded as a data object. The intermediate nodes describe the extent that the objects are proximal to each other; and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or an object and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels.

The operation of a hierarchical clustering algorithm is illustrated using the two-dimensional data set in Fig. 2.1(a). This figure depicts seven input data labeled A, B, C, D, E, F , and G . One possible dendrogram is shown in Fig. 2.1(b). With the hierarchical structure, the different clustering results can be obtained for different similarity requirements. As shown in Fig. 2.1(a), if the similarity requirement is set at *level 1*, the input data set is partitioned into two clusters, i.e., $\{A, B, C, D\}$ and $\{E, F, G\}$. However, if the similarity requirement is set at *level 2*, then the input data

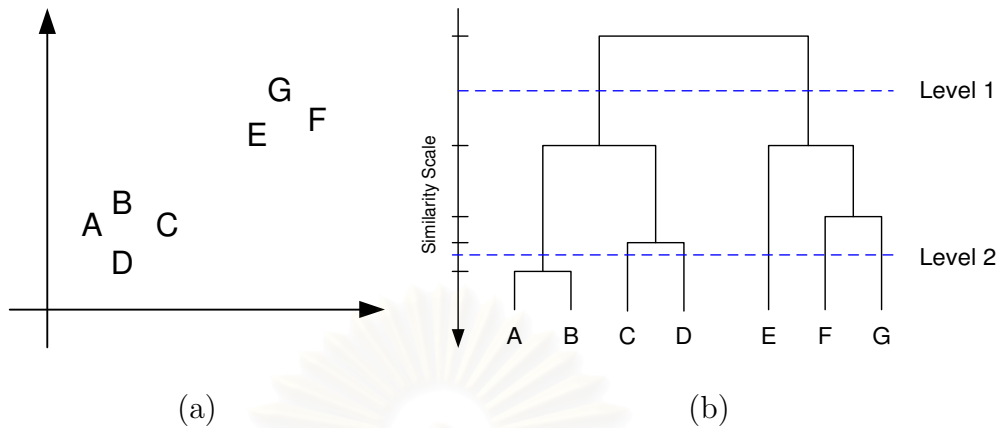


Figure 2.1: The hierarchical clustering algorithm for a data set of seven points. (a) The input data set. (b) A possible dendrogram.

set is partitioned into six clusters, i.e., $\{A, B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{F\}$, and $\{G\}$.

Most hierarchical clustering algorithms are variants of the single-link, complete-link, centroid-link, average-link, and minimum-variance algorithms. The single-link and complete-link algorithms are most popular. These two algorithms differ in the way they characterize the similarity between pair of clusters (inter-cluster distance). The widely used measures of inter-cluster distance in these algorithms are listed in Table 2.1 (m_i is the mean for cluster c_i and n_i is the number of data points in cluster c_i) [25]. In classical hierarchical clustering, two clusters are merged to form a larger cluster based on minimum inter-cluster distance criteria.

The hierarchical clustering algorithm can be summarized by the following procedure.

1. Initially, each data point forms a cluster by itself.
2. The inter-cluster distance matrix for all distinct pairs of input data is constructed as the proximity matrix.
3. The algorithm repeatedly merges the two closest clusters.
4. The output of the algorithm is a hierarchical structure which can be cut at a

Table 2.1: Types of definition of inter-cluster distance.

Definition	Inter-cluster distance
Single-link	$d_{min}(c_i, c_j) = \min_{x_i \in c_i, x_j \in c_j} \ x_i - x_j\ $
Complete-link	$d_{max}(c_i, c_j) = \max_{x_i \in c_i, x_j \in c_j} \ x_i - x_j\ $
Centroid-link	$d_{mean}(c_i, c_j) = \ m_i - m_j\ $
Average-link	$d_{ave}(c_i, c_j) = \frac{1}{(n_i n_j)} \sum_{x_i \in c_i} \sum_{x_j \in c_j} \ x_i - x_j\ $
Minimum Variance	$d_{ward}(c_i, c_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \ m_i - m_j\ $

desired similarity requirement forming the partitioned clusters.

However, if the clusters are close to one another because of the noisy data, the single-link gives uncorrectness result. As well as the complete-link clustering algorithm has problems in dealing with particular shapes.

2.2 Partitional Clustering Algorithms

Partitional clustering attempts to break a data set into k clusters such that the partition optimizes a given criterion [22]. The k -means algorithm is the best-known partitional algorithm for data clustering. It starts with the random initial k partitions and keeps reassigning the input data to clusters based on the similarity between the data point and the cluster centers until a convergence criterion is met. The outline of the k -means algorithm is given as follows:

1. Initially, k centroids are selected arbitrarily for each cluster c_i , $i \in [1, k]$.
2. Each data point assigns to the cluster whose centroid is closest to the data point.
3. Each cluster center is recalculated.

4. Step 2 and Step 3 are repeated until no data points change between clusters.

The k -means algorithm is popular because it is easy to implement. However, a major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen. The k -means algorithm fails for data in which data points in a given cluster are closer to the center of another cluster than to the center of their own cluster. This can happen in many natural clusters, for example, when cluster shapes are convex. Besides, k -means is also sensitive to noise.

2.3 Hybrid Clustering Algorithms

To overcome the problems of hierarchical and partitional clustering algorithms, several clustering methods have been proposed to combine the features of hierarchical and partitional clustering algorithms. It called the hybrid idea or two-phase clustering algorithm. In general, these algorithms first partition the input data set into small subclusters instead of using all the data points as the distinct clusters. Then, these algorithms construct a hierarchical structure based on these subclusters and these subclusters are next grouped into larger clusters.

Clustering Using Representatives (CURE) Algorithm

Algorithm CURE [10] is an improvement over the single-link clustering algorithm. In CURE, instead of using all data points to represent the clusters, a fixed number of well scattered points are chosen to represent a cluster. The scattered points capture the shape and extent of the cluster. The chosen scattered points are next shrunk towards the centroid of the cluster by a shrinking factor. These scattered points after shrinking are used as representatives of the clusters. The clusters with the closest pair of representative

points are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. CURE is capable of finding clusters of different shapes and sizes. Shrinking the scattered points towards the centroid helps CURE in avoiding the problem of noises.

A major limitation of CURE algorithm is that the merging decisions are based upon pre-defined parameters which are a shrinking factor, the number of representative points, the number of clusters. CURE algorithm can breakdown if the choice of parameters used in the merging criteria is incorrect with respect to the data set.

Cohesion-based Self-Merging (CSM) Algorithm

The CSM algorithm [24] is the recent two-phase clustering algorithm. The new similarity measure between two clusters is proposed, namely, *cohesion* based on the join ability of two clusters with density impedance to resist the effects of noises defined as Eq. 2.1.

$$similarity(c_i, c_j) = \frac{cohesion(c_i, c_j)}{impedance(c_i, c_j)^\alpha} \quad (2.1)$$

where α is the impedance factor specified by users.

In the first phase, algorithm CSM adopts the k -means algorithm to divide the input data set into m subclusters. At the beginning of second phase, it obtains the cohesions of these m subclusters produced in the first phase. Then algorithm CSM performs a single-link clustering algorithm based on cohesion to obtain the k clusters. During merging process, algorithm CSM removes all the subclusters whose sizes are less than the threshold value. The CSM algorithm is described as follows:

1. The k -means algorithm is applied on the input data set to obtain m subclusters.
2. The single-link clustering algorithm is used on the m subclusters produced in phase 1 with cohesion as the similarity measure by using Eq. 2.1.
3. Phase 2 repetitively merges two subclusters until the g clusters are obtained.

4. During phase 2, when those m subclusters are merged into m' subclusters, algorithm CSM removes all the subclusters whose sizes are less than the threshold defined as

$$size_ratio \times \frac{1}{m'} \sum |c_i|.$$

Note that $size_ratio$ and m' are two parameters specified by users.

Like CURE, algorithm CSM is sensitive to parameter settings. It is observed that when the value of m is too small, the subclusters produced in phase one may not properly partition the input data set. Thus, algorithm CSM results in an incorrect partition. On the other hand, if the input data set is partitioned into too many subclusters, algorithm CSM may also fail to partition the input data set due to the existence of many noisy subclusters. Those noisy subclusters may form a link and connect two neighboring subclusters. The *impedance factor* α also affects the correctness of the clustering results. This parameter is specified by users to control the effect of impedance. If parameter α is set too high, the CSM algorithm will make the noisy subclusters harder to join into normal subclusters. However, this may cause the normal subclusters also harder to join together. On the other hand, If parameter α is set too low, the noisy subclusters may be easy to merge into normal subclusters. Thus, algorithm CSM results in an incorrect clustering results. With proper parameter settings, algorithm CSM may give the correctness results.

2.4 Self-Organizing Map

Self-Organizing Map (SOM) is one of the most popular neural network models developed by professor Kohonen [26]. The SOM has been proven useful in many applications [27] especially data visualization and data clustering. The SOM algorithm is based on

unsupervised learning, which means that no human intervention is needed during the learning. The SOM algorithm is quite a unique kind of neural network in the sense that it constructs a topology preserving mapping from the high-dimension space onto map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimension space onto a plane. Thus, the SOM network architecture is a two-layer neural networks with one input layer and one output layer.

The SOM is a two-dimensional array of P neurons. Every neuron i of the map is associated with a d -dimensional reference vector (weight vector) $w_i = (w_{i1}, w_{i2}, \dots, w_{id})$. This has the same dimension as the input data vector. The neurons of the map are connected to adjacent neurons by a neighborhood relation, which dictates the topology, or the structure, of the map. The most common topologies in use are rectangular or hexagonal topology as illustrated in Fig. 2.2. At each training step t , an input data vector $x(t)$ is randomly chosen from the training set. Distance between $x(t)$ and all weight vectors are computed. The best matching unit, denoted by c , is the neuron with the weight vector closest to $x(t)$ such that

$$\|w_c(t) - x(t)\| \leq \|w_i(t) - x(t)\|, \quad i \in 1, 2, \dots, P \quad (2.2)$$

where $\|\cdot\|$ is the Euclidean distance.

A set of neighboring neurons of the best matching unit is denoted as N_c , which decreases its neighboring radius of the best matching unit with time. $h_{ic}(t)$ defines as

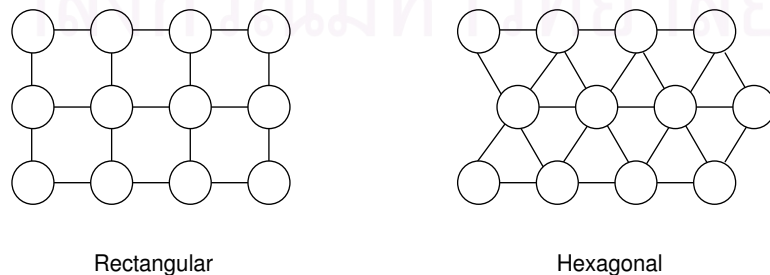


Figure 2.2: The most common topologies.

a neighborhood kernel function around the best matching unit c at time t . The kernel can be taken as a gaussian function,

$$h_{ic}(t) = \exp\left(\frac{-\|r_i - r_c\|^2}{2\sigma^2(t)}\right), \quad i \in N_c \quad (2.3)$$

where r_i is the coordinates of neuron i on the topological grid and $\sigma(t)$ is a kernel width.

After the best matching unit has been found, the weight vectors are updated. The best matching unit itself as well as its topological neighbors are moved closer to the input vector in the input space. So the weight update rule is the following

$$w_i(t+1) = w_i(t) + \alpha(t)h_{ic}(t)(x(t) - w_i(t)), \quad \forall i \in N_c \quad (2.4)$$

$$w_i(t+1) = w_i(t), \quad i \notin N_c \quad (2.5)$$

where $\alpha(t)$ is the learning rate at time t . Before the training process, the topology and the number of neuron must be fixed. The learning process of the SOM is summarized as follows:

1. The weight vectors of all neurons are initialized in a random manner.
2. One input vector x is randomly chosen from the input data set.
3. Find the best matching unit c at time step t by using the minimum Euclidean distance criterion as in Eq. 2.2.
4. Update the weight vectors of all neurons by using Eqs. 2.4 and 2.5.
5. Steps 2 to 4 are repeated until no noticeable changes in the feature map are observed.

Once the SOM algorithm has converged, the topological map computed by the algorithm displays important characteristics of the input space.

2.5 Variants of SOM

In the classical SOM algorithm, the topological map and the number of neurons are fixed from the beginning. This may lead to many experiments with different sized maps, trying to obtain the optimal results. As the number of neurons increases, the time it takes to do any operations on the map also increases. Several improved SOM and related algorithms [28, 29, 30] have been proposed in recent years to overcome the basic SOM problems.

One of the variants of SOM is *Cell-Splitting Grid* algorithm (CSG) which dynamically increases neural network [30]. The CSG network architecture is like the two-dimensional SOM architecture. It is a two-layer neural network with an input layer and an output layer. The neural weights connecting the input and output layer represent feature vectors. Furthermore, there are lateral connections among neighboring neurons.

The CSG network topological map is constrained in a square of unit length. All neurons are generated within the square. Each neuron corresponds to a square region with different size and neighboring neurons connected to form the neighboring relation. Note that the neighboring neurons also mean the direct left, right, top, bottom, top-left, bottom-left, top-right, and bottom-right neurons of one neuron. A typical neural network topological map is shown in Fig. 2.3.

The training process of CSG algorithm is like the SOM. But during the processing in the CSG algorithm, the network itself determines the growth of new neurons according to the activation level which tells how many times each neuron has been the best matching unit during training. When the activation level of the best matching unit decreases to zero, the CSG algorithm performs the cell-splitting mechanism, i.e., to delete the best matching unit and then generate four new neurons.

Weight initialization is very important at the time of cell-splitting in order to avoid

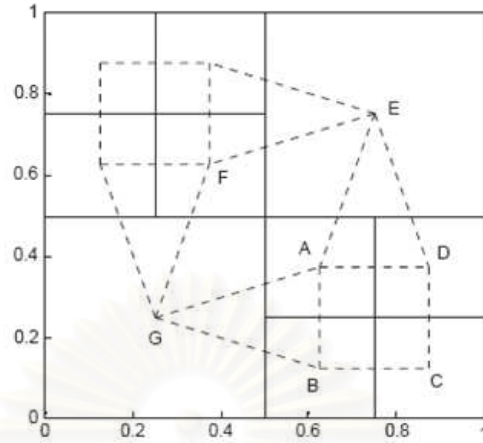


Figure 2.3: Topological map with 10 neurons at certain learning stage. A is the best matching unit, and $B, C, D, E, F,$ and G are its direct neighbors.

disorder of topology in the input space. Fig. 2.4 illustrates the weight-endowing process before and after the splitting in the two-dimensional input data. The following strategy to create an ordered topology is described.

1. When there is only one neuron at the first stage, four new neurons are generated after splitting. In order to create weights of new neurons, a vector $Random$ is introduced such that $Random$ has the same dimension as the weight vector w and satisfies $\|Random\| \ll \|w\|$. $Random$ is divided into two vectors. The first vector is A . Data in the first $\lfloor \frac{d}{2} \rfloor$ dimensions of A are the same as $Random$ ($\lfloor b \rfloor$ denotes an maximum integer less than b), but data in the rest dimensions of A is zeros. The second vector B is opposite, data in the first $\lfloor \frac{d}{2} \rfloor$ dimensions are zeros and the data in the rest dimensions are the same as $Random$. Hence, $Random$ can be expressed by $Random = A + B$. Thus, $w + A + B$, $w + A - B$, $w - A + B$, and $w - A - B$ are used to represent the new four neurons which lie on the top-right, bottom-right, top-left, and bottom-left corners in the region of the original neuron.
2. When the number of neurons is larger than 1, the condition is different because the

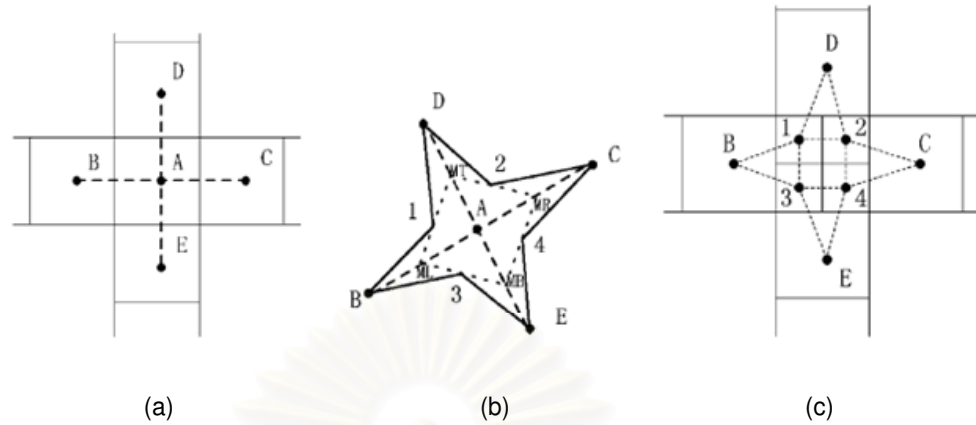


Figure 2.4: Weight initialization in the CSG algorithm for the two-dimensional input data. (a) Neuron A is to be split to generate four new neurons on the topological map. (b) The best matching unit A and the new neurons (1, 2, 3, and 4) in the two-dimensional input space before and after splitting. ML , MR , MT , and MB are middle-points between neuron A and its direct left, right, top and bottom neurons, respectively. Dashed lines ($B1$, $1D$, $D2$, $2C$, $C4$, $4E$, $E3$, and $3B$) denote the neuron connections after splitting. (c) New neurons (1, 2, 3, and 4) with new connections after splitting on the output map.

neighboring neurons obtained can be utilized. When a neuron is to be split, firstly the middle-points ML , MR , MT , and MB are computed between it and its direct left, right, top, and bottom neurons in the input space as shown in Fig. 2.4(b). If there are several direct neighbor neurons at the same direction, the middle-points between the splitting neuron and its direct neighbor neurons at that direction are averaged. After computing the middle-points, the four new neurons with weights $w_1 = \frac{ML+MT}{2}$, $w_2 = \frac{MR+MT}{2}$, $w_3 = \frac{ML+MB}{2}$, and $w_4 = \frac{MR+MB}{2}$ are generated.

Fig. 2.5 gives an example result by CSG algorithm for two-dimensional input data. However, CSG algorithm requires a full-search over the entire set of neurons to find the best matching unit. Thus, it becomes computationally impractical when the number of

neurons also increases.

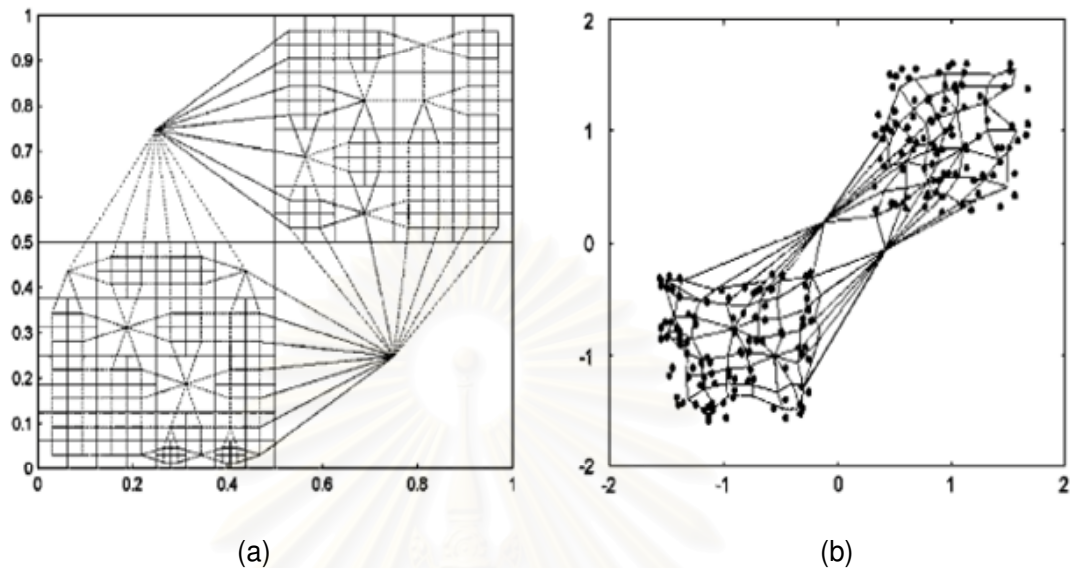


Figure 2.5: Learning by CSG algorithm. (a) Output map. (b) Input space with neuron connections.

The *Evolving Tree* (ETree) is another variant of SOM which tries to build efficient search structure to make operation faster [31, 32]. The ETree is a new kind of SOM that has been designed to scale to very large problems. The basic ETree algorithm is briefly described. A more detailed description of the basic algorithm can be found in [29]. This algorithm starts with the small ETree in Fig. 2.6(a). This example tree has a fanout of 2 for simplicity. In practice larger values are often used. The tree consists of back *leaf nodes* and white *trunk nodes*. Each node has a *prototype vector* w_i , which places in somewhere in the data space. It also has a counter b_i , which tells how many times it has been the best matching unit. A training set of data vectors are used in training one by one. Training the tree starts by finding the best matching with a greedy tree search. For every training vector x_i , the searching starts at the root node and selects the child which is closest to the training vector. This node is selected and its children are examined and until a leaf node is found. This is the best matching unit.

When the best matching units has been found, the leaf node locations are updated by using the Kohonen learning rule.

$$w_i(t+1) = w_i(t) + h_{ic}(t)(x(t) - w_i(t)) \quad (2.6)$$

The function h_{ic} defines the amount of adaptation and is a gaussian function as in SOM,

$$h_{ic}(t) = \alpha(t) \exp\left(\frac{-\|r_i - r_c\|^2}{2\sigma^2(t)}\right) \quad (2.7)$$

here, α and σ are used to control the width and time decay of the neighborhood function.

The problematic part is the function $\|r_i - r_c\|$, which tells how far apart the nodes r_c and r_i are in the SOM grid. The ETree does not form a grid so some other method is required. An equivalent metric call the *tree distance* which can be seen in Fig. 2.6(b). The idea is to calculate the amount of hops needed to get from one node to the other along the tree. In this case, five hops are needed to get from A to B . Using this distance the SOM neighborhood function can be applied.

These two steps – finding the best matching unit and updating leaf nodes – form most of the training. The third step is growing the tree. Every node has a counter that tells how many times it has been the best matching unit. When the counter reaches a certain value, called the *splitting threshold*, the node is split. That is, it is given some child nodes, thus becoming a trunk node. The child nodes' prototype vectors are initialized to their parent's value. Now the very simple basic ETree algorithm for a single training vector can be described in the following.

1. Find the best matching unit using the search tree.
2. Update the leaf node locations using the SOM training formulas substituting tree distance for grid distance.

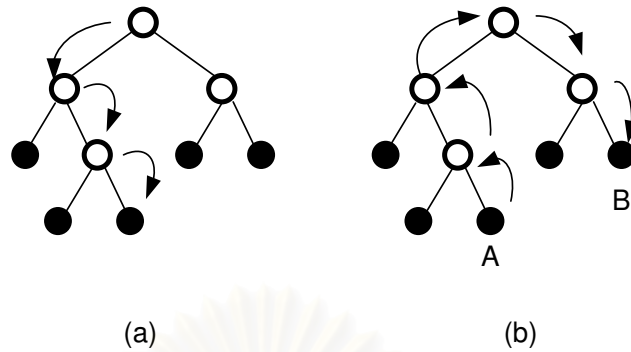


Figure 2.6: Fundamental operations of the ETree. (a) Best matching unit search. (b) Tree distance.

3. Increment the best matching unit counter.
4. If the counter reaches the splitting threshold, split the node.

This is repeated for every vectors on the training set until the system is deemed good enough. Usually this means going through the data a pre-specified amount of times.

Although ETree has been designed to make a more flexible topology, and to reduce the time consuming search for the best matching unit in large maps of the classical SOM, but ETree does not consider to preserve the correlation between the trunk node and its children. Therefore, the updating of the trunk nodes at each layer is essential. Since the search for the best matching unit is performed in the tree search manner, the neurons are likely to be dragged far away from their parents in the learning process. That means making the search for best matching units more difficult and incorrectness. Thus, both the intralayer relationship (the neighborhood relationship) and the interlayer relationship (the parent-children relationship) should be maintained.

CHAPTER III

PROPOSED METHOD

The goal of this work is to separate a set of finite N input patterns $X = \{x_1, \dots, x_i, \dots, x_N\}$, where $x_i = (x_{i1}, x_{i2}, \dots, x_{id}) \in R^d$, into the feasible number of clusters K automatically obtained from our proposed method. The new hybrid system is proposed that tries to combine the features of hierarchical and partitional clustering algorithms called “Self-Partition and Self-Merging (SPSM)”. Fig. 3.1 presents the overview of the proposed method consisted of three sequential phases.

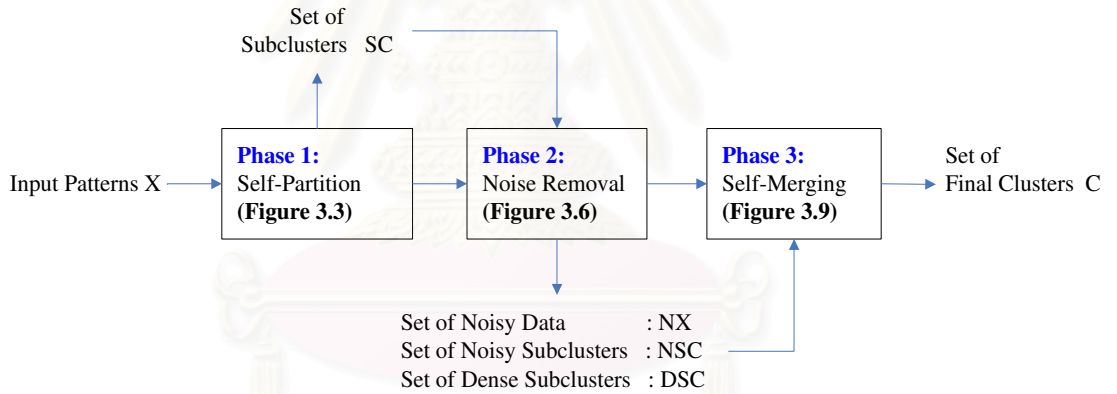


Figure 3.1: The overview of the proposed method – SPSM.

The main task of Phase 1 is to apply the proposed algorithm – Dynamic Tree-Structured Self-Organizing map (DTS-SOM) for partitioning the input data set to obtain the set of M subclusters. The DTS-SOM is a variant of SOM which is a self-creating and self-organizing algorithm designed to improve the SOM algorithm. By using DTS-SOM, the parameter settings specified by users in partition clustering algorithm e.g. the number of clusters or initial guesses, can avoid. Once the DTS-SOM performed in this phase, the number of initial subclusters is automatically obtained. The difference between the DTS-SOM algorithm and the variants of SOM are the following

- Since the network size (the number of nodes) is not pre-specified, DTS-SOM is either to make a more flexible topology for different input data set, or to reduce the computational requirements of the SOM especially the time-consuming search for the best matching unit in large maps. But CSG algorithm requires a full-search over the entire set of neurons to find the best matching unit, so it becomes computationally impractical.
- During the weight updating processing, unlike ETree algorithm, the DTS-SOM algorithm not only updates the prototype vector of the best matching unit including its direct neighbors, but also updates its ancestors to maintain the tree structure. Because all the operations are performed in the tree search manner. The updating processing at each layer is necessary. This will lead to the correctness of finding the best matching unit using the DTS-SOM training is more than the correctness of finding the best matching unit using the ETree algorithm.

In unsupervised classification problem, most of the input data set always contain *noises*. Usually, the presence of noise indicates some sort of problem. This can be a case which does not fit the model under study, or an error in measurement. Therefore, *noises* are those random points that are very different from others and do not belong to any clusters [24]. Since no priori knowledge is provided in the clustering process, it is hard to identify which data points are likely to be the noisy data. To remove such noisy data before performing Phase 1 will cause the correctness of the characteristic of data set such as shapes of data or the data distribution. Thus, instead of trying to remove the expected noisy points immediately, algorithm SPSM first partitions the input data set into several subclusters and identifies the noisy subclusters.

Therefore after Phase 1, it is noticed that some subclusters consist of only noisy data. Those noisy subclusters will affect the correctness of the subsequent merging. Besides,

it is observed that the data points in noisy subclusters are generally sparse compared to data points in dense subclusters, and the density of noisy subclusters is comparatively less than the density of dense subclusters. Fig. 3.2 shows an example of the sparse noisy subclusters that are likely to be merged with others and may become bridges between subclusters which should be separated. So the main purpose of Phase 2 is to filter out a majority of the noisy subclusters. After this Phase, the SPSM algorithm obtains the set of dense subclusters that capture the major distribution pattern in the data, the set of noisy subclusters, and some of noisy data points.

In the last Phase, an agglomerative clustering algorithm is adapted by applying it directly to the dense clusters to find the genuine clusters by repeatedly combining together these similar subclusters. In the proposed agglomerative clustering, inter-distance and intra-distance are incorporated into merging criteria. The rationale using the distances as the merging criteria is that the noisy subclusters acquired from Phase 2 have been removed, so the merging problem of the noisy subclusters connected as the bridge between two subclusters when using the distance criteria, can overcome. After performing all three phases, the input data set can be extracted the final clusters and identified the noisy data. Therefore, our proposed method can produce the feasible estimation of the number of output clusters by itself.

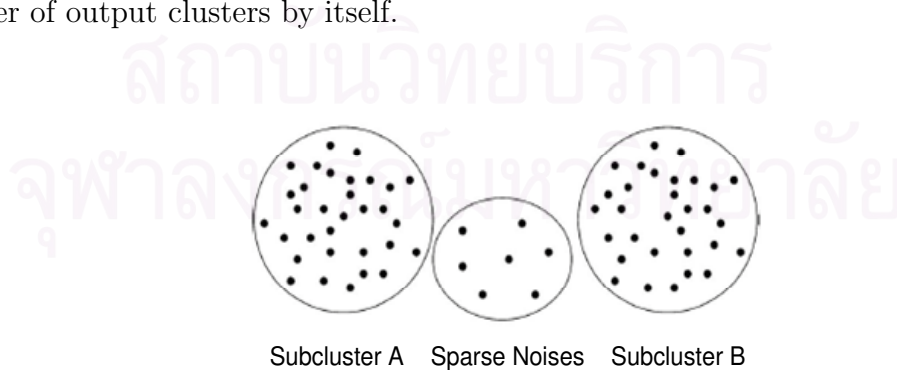


Figure 3.2: An example of a sparse noisy subcluster generated in Phase 1.

3.1 Phase 1: Self-Partition

The objective of this Phase is to partition the input data into small subcluster by using DTS-SOM algorithm as a partitioning tool. During the DTS-SOM training, the neural nodes (nodes or neurons) are arranged in a tree topology and allowed to grow when any given branch receives a lot of times being the best matching unit from the training data vectors. The search for the best matching unit and its neighbors is conducted along the tree and is therefore very efficient. The DTS-SOM has adopted technique used in CSG algorithm for weight vector initialization described in Chapter 2. After DTS-SOM training, the tree structure of DTS-SOM is produced. In our network, the prototype vectors of leaf nodes are used as the subcluster centers. For further using the information of each subcluster, then each data point is assigned to its nearest subcluster center in the tree search manner. So the process of Phase 1 has been designed into two steps as shown in Fig. 3.3. The first step applies the DTS-SOM training on the input data set to obtain the subcluster centers. Then the second step uses to identify the membership to each data point.

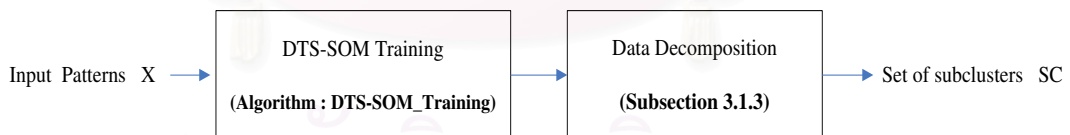


Figure 3.3: The overall steps of Phase 1.

3.1.1 Architecture of the DTS-SOM

The DTS-SOM has nodes with prototype vectors, just like the SOM. Let $w_i \in R^d$ denote the prototype or weight vector of node i . In addition, each node in the network has an initial value τ_i as the activation level τ . This activation level is the counter which tells how many times each node has been the best matching unit during training.

Furthermore, there are lateral connections among neighboring neurons. Note that the neighboring neurons also mean the direct left, right, top, bottom, top-left, bottom-left, top-right, and bottom-right neurons of one neuron that is the same definition as the CSG algorithm.

3.1.2 Training of the DTS-SOM

The DTS-SOM algorithm starts by taking a single neuron and placing it at a suitable place in the data space. An obvious suitable choice is the center of mass of the data cloud. Then, the node is split to generate four new neurons and their activation levels are initialized including the lateral connections. This means that a pre-determined amount of new nodes is created and then the algorithm marks them as the children of the split node. The weight vectors are initialized by using the weight initialization method proposed in the CSG algorithm. For the next training vector, the best matching unit is chosen among the children nodes in a manner described later. Now a tree structure is produced with four leaf nodes and one root node as the initialized DTS-SOM network. Once a leaf node is activated as the best matching unit, the activation level τ of such neuron decreases by a constant value. This process continues until τ of one leaf node becomes zero and the neuron is split to generate its four offspring neurons. Then, the activation levels are set to the new generated neurons. After the initial activation levels (τ_i) are given to the new neurons, the activation levels of all neurons are increased by $\Delta\tau > 0$ to slow down the splitting rate. Thus, a tree is formed recursively by the training algorithm. During the training process, the DTS-SOM forms an elastic network that folds onto the data cloud. The algorithm controls the network so that it tries to approximate the density of the data. The neurons drift to the areas where the density of the input data is high. Eventually, only few neurons lie in the areas where the input

data is sparse.

Now the algorithms: how to find the best matching unit in a tree and how to train the tree structure are described. To illustrate this process, a larger tree structure is assumed at certain learning stage, which can be seen in Fig. 3.4. In the DTS-SOM, every internal nodes (white nodes) have four children (black nodes). Finding the best matching unit is a top-down process. The searching starts with the root node, then its children are examined. The DTS-SOM finds the node whose prototype vector is closest to the training vector. If that node is a leaf node, then it is the best matching unit. If it is not, its children are examined in turn and the closest one of them is chosen. This is repeated until a leaf node is found. Thus, the DTS-SOM's internal nodes work as a hierarchical search tree for the leaf node.

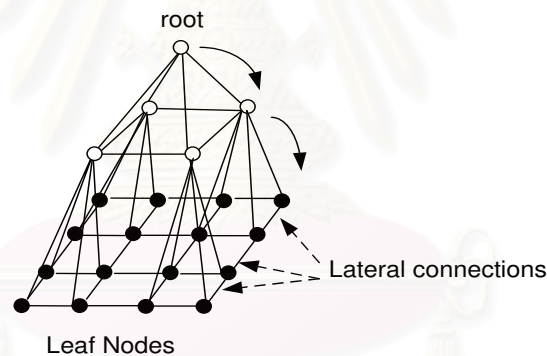


Figure 3.4: Tree search operation of the DTS-SOM: how the best matching unit is found. White and black nodes are denoted the internal nodes and the leaf nodes, respectively. The arrows show tree search path to find the best matching unit.

When the best matching unit has been examined, it is time to update the prototype vector of the best matching unit, but also those of its direct neighbors, towards the training vector. The Kohonen learning rule [26] is used to update node weights $w_i(t)$.

towards the training vector $x_i(t)$:

$$w_c(t+1) = w_c(t) + \alpha[x_i(t) - w_c(t+1)] \quad (3.1)$$

$$w_b(t+1) = w_b(t) + \alpha h_{bc}[x_i(t) - w_b(t+1)] \quad (3.2)$$

Here, c is the index of the best matching unit and b is index of all the direct neighbor neurons of the best matching unit c as before mentioned. The function h_{bc} defines the neighborhood function and a common choice for the neighborhood function is Gaussian neighborhood as in SOM.

$$h_{cb} = \exp\left(\frac{-\|r_c - r_b\|^2}{2\sigma^2}\right) \quad (3.3)$$

The vectors r_c and r_b give the locations of nodes c and b on the SOM regular grid that has an unit length between each node. The parameter α defines the learning rate, and σ gives the width of the gaussian kernel.

The updating function at each layer is essential. Since all the operations are performed in the tree search manner described above, the neurons are likely to be dragged far away from their parents in the learning process. If there is no provision to preserve the correlation between the parent and its children neurons, the tree structure will be destroyed, making the search for best matching units more difficult and the overall distortion enlarged. Therefore, both the intralayer relationship (the neighborhood relationship) and the interlayer relationship (the parent-children relationship) are maintained. To keep the neurons of the same family close during the entire training process, the parent neurons (internal nodes) will only be updated if their children are also updated with the average value of all their children weight vectors as shown in Eq. 3.4,

$$w_j(t+1) = \frac{1}{nc_j} \sum_{\forall k} w_k(t) \quad (3.4)$$

where nc_j is the number of children of neuron j and k is index of all children of neuron j . Fig. 3.5 illustrates an example the DTS-SOM learning for the two-dimensional input data at certain learning stage. Fig. 3.5(b) depicts the dynamic topology of the leaf neurons in the input space according to the tree structure as shown in 3.5(a).

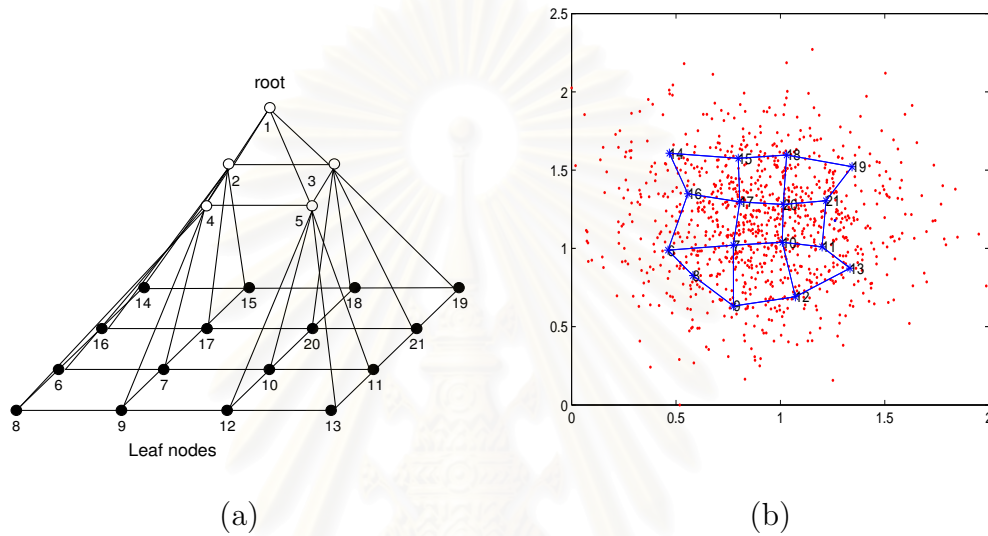


Figure 3.5: An example of the DTS-SOM learning. (a) The tree structure. (b) Input space with the leaf neuron connections. A number shows the neuron index.

The overall idea of the DTS-SOM algorithm is shown in **Algorithm DTS-SOM**. **Training.** This is repeated for every vector on the training set until the system is deemed good enough. At the end of each epoch, the size of the tree is examined. So in our system, the size of the tree nodes in the current and the previous epochs are compared. If their difference is less than a small threshold value, then the training is completed.

In our simulations, the DTS-SOM training makes use the following parameters.

- The learning rate α must be rather small, but would not be decreased to zero. Normally, the learning rate α is set to a value less than 1 and the width of neighborhood function σ is set to a value typically between 0.8-0.99 [26, 33].

- The initial activation level τ_i affects the growing rate of the network. If τ_i is too small, the network size increases without learning enough information and finally will not be able to well represent the topology of the input data. Usually, τ_i is chosen to $0.05N \sim 0.5N$, where N is the number of input data.
- At each splitting stage, the activation levels of all neurons are increased by $\Delta\tau$. This causes the splitting rate slower and slower. Usually $\Delta\tau$ can be set to $\Delta\tau = (0, 1]$ of the activation level τ_i .
- The training is terminated if the tree has grown less than a specified amount such as the size of the tree nodes in the current is grown slower than 5% of the size of tree nodes in the previous epochs.

Algorithm DTS-SOM_Training

Input: input data $x_j(t)$ at time t .

Output: the tree structure of DTS-SOM network.

Begin

1. Initialize the DTS-SOM network as described at the beginning of Sec. 3.1.2.
2. **Repeat**
3. Select an input datum $x_j(t)$ randomly from input space.
4. Find the best matching neuron c using the tree search such that $\|w_c(t) - x_j(t)\| \leq \|w_i(t) - x_j(t)\|, \forall i$.
5. Update weight vectors of the best matching unit c using Eq. 3.1, and neurons in N_c using Eq. 3.2, where N_c is the set of the direct neighbor neurons of the best matching unit c .
6. Update their parents layer by layer using Eq. 3.4.
7. Decrease the activation level of the best matching unit c by 1.

If the activation level of the best matching unit c decreases to zero **Then**

Generate four new neurons of original neuron c .

10. Initialize the new weights and the activation level of the new generated neurons.
11. **Endif**
12. Increase the activation levels of all neurons by $\Delta\tau$.
13. **Until** the tree has grown less than a small threshold value.

End

3.1.3 Data Decomposition

After DTS-SOM training, the tree structure of DTS-SOM network is obtained which is consisted of the number of leaf nodes and the number of internal nodes. In our network, the leaf nodes are the most important part because the prototype vectors of leaf nodes are represented the subcluster centers. Therefore, the number of leaf nodes is equivalent to the the number of subclusters. Next each data point is mapped to its nearest subcluster center in the tree search manner, and recalculate the subcluster centers with the average of all data points in each subcluster. After this stage, the input data space is partitioned into small subclusters without prior knowledge. The set of subclusters $SC = \{sc_1, \dots, sc_j, \dots, sc_M\}$, where M is the number of subclusters, is defined.

3.2 Phase 2: Noise Removal

The purpose of Phase 2 is to identify which subclusters are the core subclusters and discard the noisy subclusters. As mentioned above, the noisy subclusters will affect the correctness of the agglomerative clustering algorithm (Phase 3). Note that the

data points of noisy subclusters are generally sparse compared to data points in dense subclusters, and the density of noisy subcluster is comparatively less than the density of dense subcluster. So, the main task is to find the density threshold for separating the set of subclusters SC into two groups which are the set of dense subclusters $DSC = \{dsc_1, \dots, dsc_j, \dots, dsc_{M'}\}$, and the set of noisy subclusters $NSC = \{nsc_1, \dots, nsc_k, \dots, nsc_{M''}\}$, where M' and M'' are the number of dense subclusters and noisy subclusters, respectively.

In order to quantify the meaning of sparseness, the volume and the density of the subcluster are introduced. The volume of each subclusters sc_j is measured as the multiple of the data eigen-axis length. Since the square roots of eigenvalue of a covariance matrix can be treated as the radius of a subcluster. So the volume and the density of a subcluster can be defined as follows:

$$sc_j_volume = \sqrt{\lambda_1} \times \sqrt{\lambda_2} \times \dots \times \sqrt{\lambda_d} \quad (3.5)$$

$$sc_j_density = \frac{|sc_j|}{sc_j_volume} \quad (3.6)$$

where $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_d}$ are the length of each axis in d -dimensional data space. sc_j_volume and $sc_j_density$ are the volume and the density of any subclusters sc_j , respectively. $|sc_j|$ is the number of data points in subcluster sc_j .

The whole process of noise removal is composed of three steps as shown in Fig. 3.6. Since the main process is to determine the density threshold for separating the subclusters, the first step is to compute the density of each subclusters. Before processing density computation, some deviated points of each subcluster are discarded because those points will affect the correctness of density computation and further merging process. Next step is used to separate the set of subclusters SC into noisy subcluster set NSC and dense subcluster set DSC based on the density threshold. In the last step, some noisy subclusters will be verified because some dense subclusters may be removed too

much from the previous step.

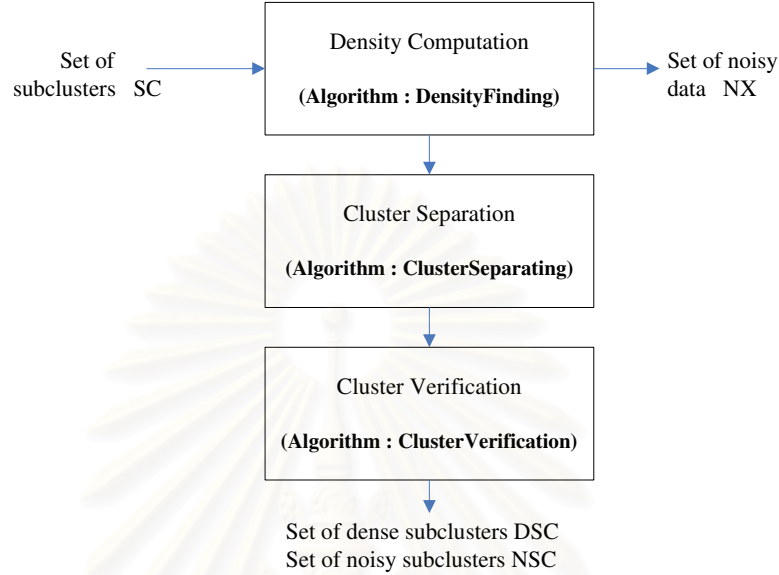


Figure 3.6: The overall steps of Phase 2.

3.2.1 Density Computation

Before finding the subcluster density, it is observed that the expected dense subclusters have some deviated points, and such those points will affect the correctness of density computation. Thus, all the data points in each subcluster are examined. If the points are outside the hyperbox of eigenvalue side length along the eigenvector directions centered at the mean of cluster, such points are removed to the set of noisy data $NX = \{nx_1, \dots, nx_i, \dots, nx_{N'}\}$, where N' is the number of noisy points. Fig. 3.7 shows an example of a point x_i which is outside the hyperbox along the eigenvectors X' and Y' directions. The square roots of eigenvalues λ_1 and λ_2 of a covariance matrix are set as the radii of the data set in the hyperbox, and r is the vector obtained from data vector x_i and the subcluster center $X0$. So, if the scalar projection of r onto X' which is the length of the segment $X0A$ is greater than $\sqrt{\lambda_1}$, then such data point x_i is discarded to

the set of noisy data NX . Thus, this process can remove some deviated data points in the subclusters but also can eliminate some noisy subclusters. So the overall algorithm of finding subcluster density is described in **Algorithm DensityFinding**. This stage produces the set of noisy data NX , the set of new subclusters SC without the deviated points, and also the densities of new subclusters SC .

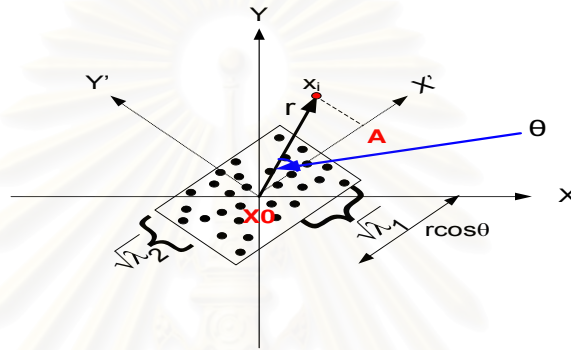


Figure 3.7: An example of a point x_i is outside the hyperbox.

Algorithm DensityFinding

Input: the set of subclusters SC .

Output: (1) the set of noisy data NX , (2) the set of new subclusters SC without the deviated points, and (3) density values of new subclusters SC .

Begin

1. **For** each subcluster sc_j **Do**
2. **If** $|sc_j|$ is greater than 1 **Then**
3. Compute the eigenvalues λ_d and the eigenvectors v_d of subcluster sc_j .
4. Compute the volume of subcluster sc_j using Eq. 3.5.
5. **For** each point $x_i \in sc_j$ **Do**
6. Compute $r = \|X_{O_{sc_j}} - x_i\|$.
7. **For** each eigenvector v_d **Do**
8. Compute angle θ_d between the eigenvector v_d and the point x_i .

```

9.          Compute  $X0_{sc_j}A = r \times \cos(\theta_d)$ .
10.         If  $X0_{sc_j}A$  is greater than  $\sqrt{\lambda_d}$  Then
11.             Remove the point  $x_i$  to the set of noisy data  $NX$ .
12.             Break.
13.         Endif
14.     Endfor
15. Endfor
16. If  $|sc_j|$  is greater than 1 Then
17.     Compute the density of subcluster  $sc_j$  using Eq. 3.6.
18. Else
19.     Remove the point  $x_i$  to the set of noisy data  $NX$ .
20.     Set the density of subcluster  $sc_j$  to zero.
21. Endif
22. Else
23.     Remove the point  $x_i$  to the set of noisy data  $NX$ .
24.     Set the density of subcluster  $sc_j$  to zero.
25. Endif

End

```

3.2.2 Cluster Separation

The purpose of this stage is to separate the set of subclusters SC without the deviated data points into two classes which are the set of dense subclusters DSC and the set of noisy subclusters NSC based on the density threshold value denoted as $threshold_{density}$. Then the density values of all subclusters are sorted in ascending order and arranged them by logarithm of density as shown in Fig. 3.8(a). From the result of density

arrangement, the density values can be categorized into four types which are type I – the lowest density, type II – the lower density, type III – the medium density, and type IV – the high density. In type III, there is the most number of density values. Moreover, this can guarantee that type I and type IV are actually density of noisy subclusters and dense subclusters, respectively. So the subclusters sc_j whose density values are in the density range of type I, can be set to the set of noisy subclusters NSC . In the same way, the subclusters sc_j whose density values are in the density range of type IV, can be set to the set of dense subclusters DSC . Then, only density values in types II and III can be used for finding the density threshold value. Fig. 3.8(b) shows the density values for types II and III.

Thus, the method to determine the density threshold value is introduced based on the coefficient of variation (CV) which indicates the relative amount of dispersion of the data [34]. In this dissertation, each subcluster density is used as the interesting data to compute CV . If $\sigma_{set1_density}$ is the standard deviation of the first set of subcluster density values and $set1_density_{mean}$ is its mean, then

$$CV_{set1} = \frac{\sigma_{set1_density}}{set1_density_{mean}} \quad (3.7)$$

From Fig. 3.8(b), it is noticed that the density values in the region expected to be noisy subclusters rather spread. Since the density values of the noisy subclusters are varied, so the threshold for separating the density values into two sets is the density such that the density values of the first set is much varied than another set based on CV . The density that makes the CV of the first set (CV_{set1}) is greater than the CV of the second set (CV_{set2}) is set as the density threshold value. Thus, this density threshold are used to remove the noisy subclusters. But it is possible that the noisy subclusters is still remaining in the second set. These noisy subclusters need to remove more. Then the subclusters are cut off when their density values are less than the average of the

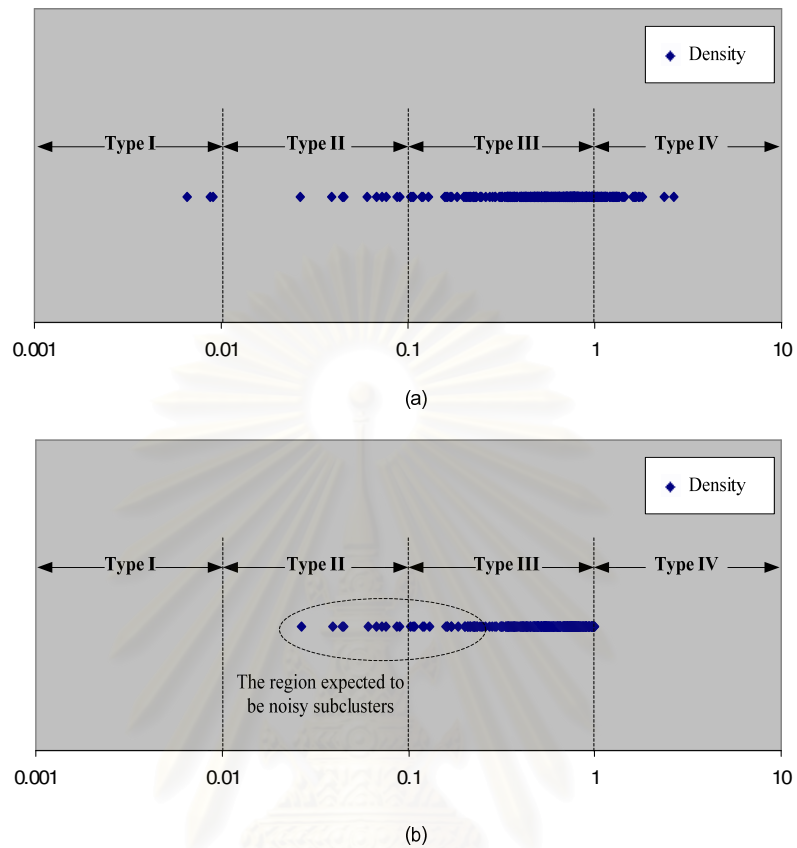


Figure 3.8: An example of the density arrangement. (a) Four types of sorted density values. (b) The elliptic region denoted by dashed line shows the expected area which may be the noisy subclusters.

density values with their standard deviation in the second set defined as $density_{mean}$ and $density_{std}$, respectively. The overall algorithm is following.

Algorithm ClusterSeparating

Input: (1) the set of subclusters SC without the sparse data and (2) sorted density values of types II and III.

Output: (1) the set of dense subclusters DSC and (2) the set of noisy subclusters NSC .

Begin

1. Set $index$ to 1.
2. Set density threshold $threshold_{density}$ to $sc_{index-density}$.
3. Define $set_1 = \{sc_i-density | i = 1 \dots index\}$.
3. Define $set_2 = \{sc_j-density | j = index + 1 \dots k\}$,
 where k is the number of densities in types II and III.
4. Compute CV_{set1} and CV_{set2} using Eq. 3.7.
5. **While** $CV_{set1} < CV_{set2}$ **Do**
6. Increase $index$ by 1.
7. Set density threshold $threshold_{density}$ to $sc_{index-density}$.
8. Define $set_1 = \{sc_i-density | i = 1 \dots index\}$.
9. Define $set_2 = \{sc_j-density | j = index + 1 \dots k\}$.
10. Compute CV_{set1} and CV_{set2} .
11. **Endwhile**
12. **For** each subcluster sc_j **Do**
13. **If** $sc_j-density$ is less than $threshold_{density}$ **Then**
14. Set subcluster sc_j to the set of noisy subclusters NSC .
15. **Else** Set subcluster sc_j to the set of dense subclusters DSC .
16. **Endif**
17. **Endfor**
18. Compute $density_{mean}$ and $density_{std}$ of the dense subcluster set DSC .
19. **For** each dense subclusters dsc_j **Do**
20. **If** $dsc_j-density$ is less than $(density_{mean} - density_{std})$ **Then**
21. Set subcluster dsc_j to the set of noisy subclusters NSC .
22. **Endif**
23. **Endfor**

End

3.2.3 Cluster Verification

As the results of the previous step, it is possible that too many subclusters are removed. This will affect to the further merging process. So some removed subclusters in NSC should be recovered back to the set of dense subclusters DSC because the center distance and the density of some noisy subclusters are near to the dense subclusters. If (1) the distance between the center of the noisy subcluster nsc_k and its nearest center of the dense subcluster dsc_j is very close, and (2) the density value of the noisy subcluster nsc_k is near to the density value of the dense subcluster dsc_j , such noisy subcluster nsc_k is recovered back to be the member of the set of dense subclusters DSC . The center distance $D0$ of any two subclusters is defined as in Eq. 3.8.

$$D0 = \|X0_1 - X0_2\| \quad (3.8)$$

where $X0_1$ and $X0_2$ are the subcluster centers of any first and second subclusters, respectively. The subcluster verification process is as follows:

Algorithm ClusterVerification

Input: (1) the set of dense subclusters DSC and (2) the set of noisy subclusters NSC .

Output: (1) the set of new dense subclusters DSC and (2) the set of new noisy subclusters NSC .

Begin

1. Compute $D0_mean$ and $D0_std$ between each center of dense subclusters dsc_j and the centers of its neighboring dense subclusters.
2. Compute $density_mean$ and $density_std$ of the noisy subcluster set NSC .
3. **For** each dense subclusters dsc_j **Do**

4. **For** each its neighboring noisy subclusters nsc_k **Do**
 5. Compute center distance $D0_{dsc_j nsc_k}$ between subclusters dsc_j and nsc_k .
 6. **If** ($D0_{jk}$ is less than $D0_mean - D0_std$) and
 ($nsc_k.density$ is greater or equal than $density_mean + density_std$) **Then**
 7. Recover nsc_k to the set of the dense subclusters DSC .
 8. **Endif**
 9. **Endfor**
 10. **Endfor**
- End**

3.3 Phase 3: Self-Merging

In this section, the details of our Self-Merging process are described. This algorithm has adapted from an agglomerative clustering algorithm by applying directly to the set of dense subclusters DSC . This process uses the distance metrics as the merging criteria. The intra-distances $D1$ and $D2$ are measured as the following.

$$D1 = \frac{1}{|dsc_j|} \sum_{i=1}^{|dsc_j|} (MIN_{\forall x_k \in dsc_j} \|x_i - x_k\|) \quad (3.9)$$

$$D2 = MAX_{\forall x_i \in dsc_j} (MIN_{\forall x_k \in dsc_j; x_i \neq x_k} \|x_i - x_k\|) \quad (3.10)$$

$D1$ is the average of the minimum pairwise distances within a subcluster. $D2$ is the maximum distance selected from the minimum pairwise distances within a subcluster. Next between two subclusters, the inter-distances for measuring their closeness are introduced.

$$D3 = MIN_{\exists x_i \in dsc_j} (MIN_{\exists x_k \in dsc_l} \|x_i - x_k\|) \quad (3.11)$$

$$D4 = MAX_{\exists x_i \in dsc_j} (MIN_{\exists x_k \in dsc_l} \|x_i - x_k\|) \quad (3.12)$$

$D3$ is the minimum distance chosen from the minimum pairwise distances between two subclusters, and $D4$ is the maximum distance selected from the minimum pairwise distances between two subclusters. For computing inter-distances in Eqs. 3.11 and 3.12, it is not necessary to use all the data points of the subcluster. Only the data points in the partial region connecting between two subclusters can be used.

In this Phase, the algorithm consists of the multiple sequential merging steps as depicted in Fig. 3.9. First, it starts to merge between the subclusters and their neighboring subclusters which are close to each other into the same cluster. So each cluster contains the set of dense subclusters. In step 2, since the connections of the neighboring subclusters are not reachable to all dense subclusters, there are still some adjacent clusters which can be joined together. At this stage, all possible adjacent clusters are considered for merging to form the larger clusters. After performing both steps, there exists some small clusters not being merged. If the further merging process is performed, it is possible that those small clusters may be incorrectly merged to one of the large clusters. So the large clusters will be temporarily removed and then small clusters are considered for the further merging. The last step is used for refinement the clusters to obtain the final clusters. The set of clusters $C = \{c_1, \dots, c_g, \dots, c_K\}$, where K is the number of clusters, are automatically produced by our proposed method.

3.3.1 Neighboring Merging

At first, the amount of data points nx_i of the dense subclusters dsc_j are recovered from the set of noisy data NX . Because all these data points nx_i have been removed from the Phase 2 and they can assist the merging process to satisfy the merging conditions. So the data points nx_i will be recovered in the partial region connecting the two dense subclusters which are close to each other based on the threshold value. This threshold

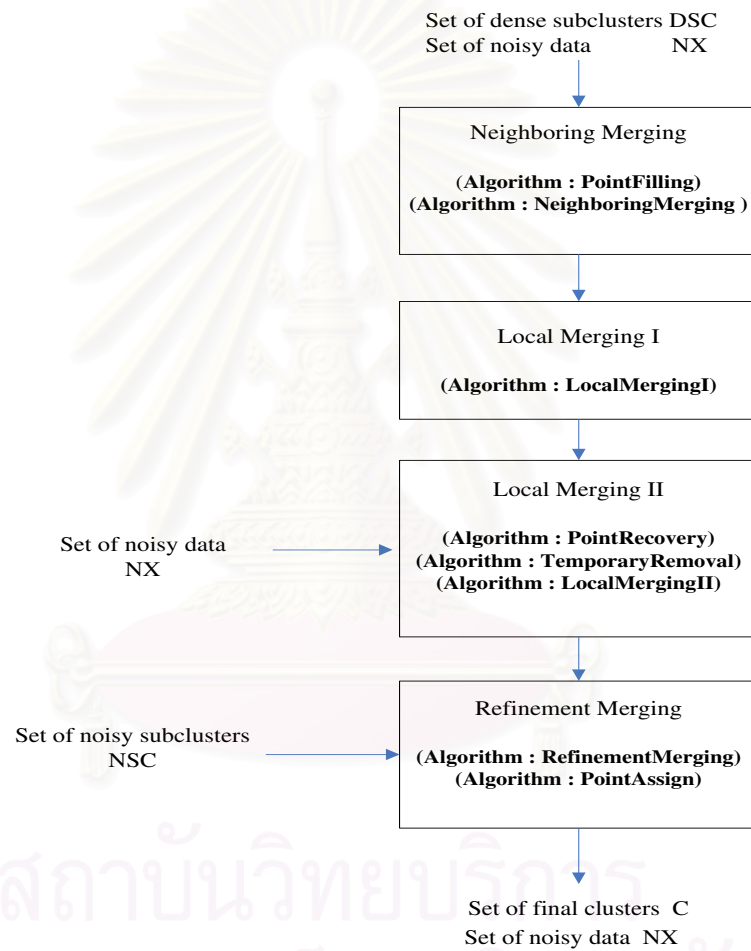


Figure 3.9: The overall steps of Phase 3.

value can be defined as the average center distances between the centers of dense subclusters dsc_j to all centers of their neighboring dense subclusters denoted by $D0_mean$ and $D0_std$ as used in **Algorithm ClusterVerification**. The partial regions are defined as shown in Fig. 3.10. Fig. 3.10(a) occurred when the sides of two subclusters are not overlapping but either sides of two subclusters in Fig. 3.10(b) are overlapping. The terms of overlapping sides used in **Algorithm PointFilling** are categorized into four cases. First, the notations are defined in the following and Fig. 3.11 shows the locations

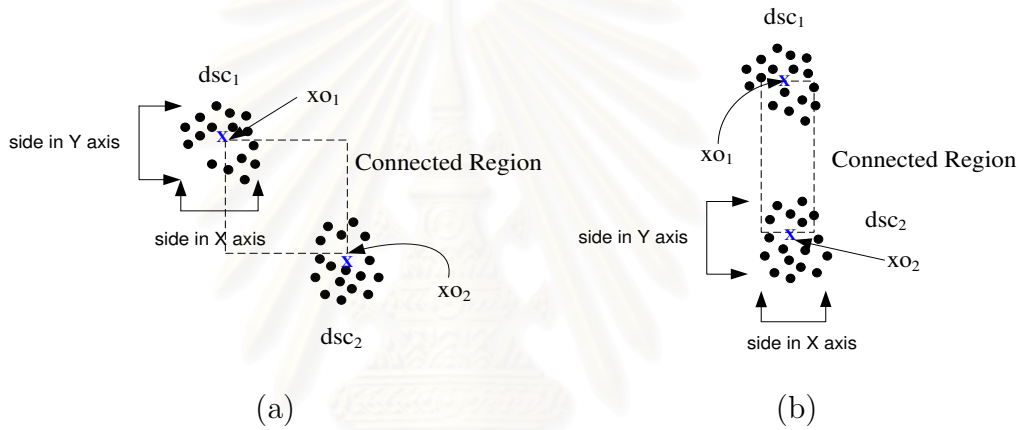


Figure 3.10: An example of the partial regions (connected regions). (a) The partial region of non-overlapping subclusters. (b) The partial region of overlapping subclusters.

of the notations.

dsc_j_min : the minimum range of each component of the data points in subclusters dsc_j ,

where $dsc_j_min = \{dsc_{j1_min}, dsc_{j2_min}, \dots, dsc_{jd_min}\}$.

dsc_j_max : the maximum range of each component of the data points in subclusters dsc_j ,

where $dsc_j_max = \{dsc_{j1_max}, dsc_{j2_max}, \dots, dsc_{jd_max}\}$.

Therefore,

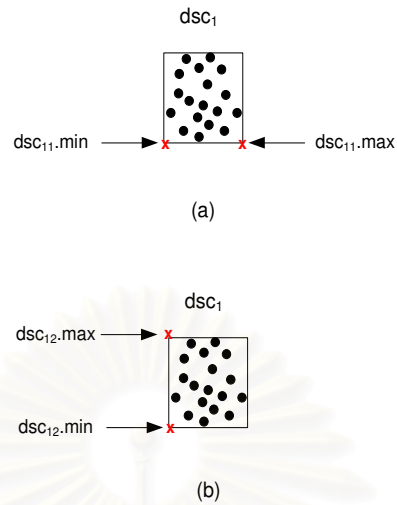


Figure 3.11: An example of the ranges of each component for two-dimensional data space. (a) The minimum and maximum positions of the first component. (b) The minimum and maximum positions of the second component.

case (1) : if $[(dsc_{jd_min} \geq dsc_{ld_min}) \text{ and } (dsc_{jd_min} \leq dsc_{ld_max})]$ and
 $[(dsc_{jd_max} \geq dsc_{ld_min}) \text{ and } (dsc_{jd_max} \leq dsc_{ld_max})]$

case (2) : if $[(dsc_{ld_min} \geq dsc_{jd_min}) \text{ and } (dsc_{ld_min} \leq dsc_{jd_max})]$ and
 $[(dsc_{ld_max} \geq dsc_{jd_min}) \text{ and } (dsc_{ld_max} \leq dsc_{jd_max})]$

case (3) : if $[dsc_{ld_min} < dsc_{jd_min}]$ and
 $[(dsc_{ld_max} \geq dsc_{jd_min}) \text{ and } (dsc_{ld_max} < dsc_{jd_max})]$

case (4) : if $[(dsc_{ld_min} > dsc_{jd_min}) \text{ and } (dsc_{ld_min} \leq dsc_{jd_max})]$ and
 $[dsc_{ld_max} > dsc_{jd_max}]$

Algorithm PointFilling

Input: (1) the set of noisy data NX and (2) the set of dense subclusters DSC .

Output: the set of dense subclusters DSC with some filled data points.

Begin

1. **For** each dense subcluster dsc_j **Do**
2. **For** each neighboring dense subcluster dsc_i of dense subcluster dsc_j **Do**
3. Compute the center distance $D0_{dsc_j dsc_i}$.
4. **If** $D0_{dsc_j dsc_i}$ is less than $(D0_mean - D0_std)$ **Then**
5. **For** each dimension d **Do**
6. **If** case (1) **Then** $overlapRange_d = [dsc_{jd_min} \ dsc_{jd_max}]$.
7. **Elseif** case (2) **Then** $overlapRange_d = [dsc_{id_min} \ dsc_{id_max}]$.
8. **Elseif** case (3) **Then** $overlapRange_d = [dsc_{jd_min} \ dsc_{id_max}]$.
9. **Elseif** case (4) **Then** $overlapRange_d = [dsc_{id_min} \ dsc_{jd_max}]$.
10. **Else**
11. **If** $(X0_{jd} < X0_{id})$ **Then** $overlapRange_d = [X0_{jd} \ X0_{id}]$.
12. **Else** $overlapRange_d = [X0_{id} \ X0_{jd}]$.
13. **Endif**
14. **Endif**
15. **Endfor**
16. Fill points nx_i in the range of $overlapRange$.
17. **Endif**
18. **Endfor**
19. **Endfor**

End

Then, the dense subclusters dsc_j and their neighboring subclusters dsc_l are merged if these dense subclusters are close to each other. First, the algorithm chooses one of the subclusters dsc_j which has the highest density as the starting seed cluster c_g . This seed grows to merge the neighboring subclusters dsc_l if the minimum inter-distance $D3_{c_g dsc_l}$ is in the range of the maximum intra-distance chosen between $D2_{c_g}$ and $D2_{dsc_l}$. This self-growing process successively merges the neighboring subclusters together until the stopping rule is satisfied. Only some data points in the partial region of the boundary subclusters are used for measuring the inter-distance. Fig. 3.12 shows the partial region denoted as same as in Fig. 3.10. The gray dots in the connected region are used for computing the inter-distance. The details of neighboring merging are described in the following.

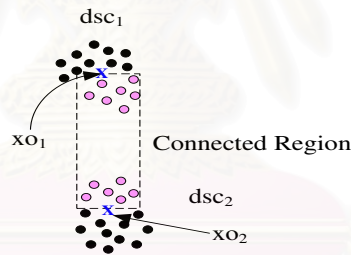


Figure 3.12: An example of the partial region (connected region). The gray dots in the connected region are used for computing the inter-distance.

Algorithm NeighboringMerging

Input: the set of dense subclusters DSC with some filled data points.

Output: the set of clusters C .

Begin

1. Compute intra-distance $D2_{dsc_j}$ of each subcluster dsc_j .
2. Set $g = 1$.
3. **While** all subclusters dsc_j are not merged **Do**

4. Choose the subcluster dsc_j which is the highest density.
5. Set the subcluster dsc_j to cluster c_g .
6. **While** the number of subclusters in c_g does not changed **Do**
7. **Repeat**
8. Compute the center distance $D0_{dsc_j dsc_l}$ between subclusters dsc_j and dsc_l .
9. **If** $D0_{dsc_j dsc_l}$ is less than $(D0_mean - D0_std)$ **Then**
10. Compute inter-distance $D3_{c_g dsc_l}$ using Eq. 3.11.
11. **If** $D3_{c_g dsc_l}$ is less or equal than $Max(D2_{c_g}, D2_{dsc_l})$ **Then**
12. Set dsc_l to the same cluster c_g .
13. **Endif**
14. **Endif**
15. **Until** there are no neighboring subclusters dsc_l of subclusters dsc_j in c_g merged.
16. Compute intra-distance $D2_{c_g}$ of cluster c_g using Eq. 3.10, $\forall x_i \in c_g$.
17. **Endwhile**
18. Increase g by 1.
19. **Endwhile**

End

3.3.2 Local Merging I

From the merging results in step 1, there exists some clusters not merged together although such clusters are very close to each other. The reason that any two adjacent clusters cannot join together is that there may be no the neighboring connections among the subclusters between two clusters. Thus, the objective of this stage is to

more aggregate the closest clusters based on $D1$ and $D3$. This step starts with selecting the cluster that has the highest number of subclusters as the starting seed cluster c_g . This seed grows to merge the neighboring clusters c_l that have similarity distances to the larger clusters. If the minimum inter-distance $D3_{c_g c_l}$ is in the range of the average intra-distance chosen between $D1_{c_g}$ and $D1_{c_l}$, then clusters c_g and c_l are joined together. Since each cluster consists of the set of subclusters, it is not necessary to use the data points of all the subclusters for computing the inter-distance. So this distance can be measured by using only the data points of the boundary subclusters as defined before. The algorithm of this stage is described below.

Algorithm LocalMergingI

Input: the set of cluster C .

Output: the set of new cluster C .

Begin

1. Compute average intra-distance $D1_{c_g}$ of each cluster c_g using Eq. 3.9, $\forall x_i \in c_g$.
2. **Repeat**
3. Choose cluster c_g which has a highest number of subclusters as the starting seed.
4. **Repeat**
5. **For** each cluster c_l **Do**
6. **If** $g \neq l$ **Then**
7. Compute inter-distance $D3_{c_g c_l}$ using Eq. 3.11, $\exists x_i \in (c_g \text{ and } c_l)$.
8. **If** $D3_{c_g c_l}$ is less or equal than $Max(D1_{c_g}, D1_{c_l})$ **Then**
9. Remove cluster c_l to cluster c_g .
10. **Endif**

11. **Endif**
12. **Endfor**
13. Compute new average intra-distance $D1_{c_g}$ of cluster c_g using Eq. 3.9,

$$\forall x_i \in c_g.$$
14. **Until** there are no clusters c_l merged to cluster c_g .
15. **Until** there are no clusters c_g merged.

End

3.3.3 Local Merging II

As before mentioned, there exists some small clusters have not been merged after process two steps. If the further merging process is performed, it is possible that those small clusters may be incorrectly merged to one of the large clusters. So the large clusters will be temporarily discarded and then all possible small clusters are joined together. The small clusters c_{small} are defined as the clusters which have the least number of subclusters in the highest frequency. The **Algorithm TemporaryRemoval** is used for temporary removal the large clusters based on the threshold value. Once the number of data points for each cluster is sorted in descending order, the threshold value can be set as the number of data points such that the difference between the number of data points in clusters c_g and c_{g+1} denoted by $diff_g$ is maximum. Before performing this algorithm, some data points nx_i which are the member of the set of subcluster DSC are recovered from the set of noisy data NX . It dues to these points can help the subsequent merging more correctness. The distance definition is defined for further using.

$$D5 = MIN_{\exists x_j \in \forall c_g} \|nx_i - x_j\| \quad (3.13)$$

$$idx = arg_g MIN_{\exists x_j \in \forall c_g} \|nx_i - x_j\| \quad (3.14)$$

$D5$ is the point inter-distance which is the minimum pairwise distance from any noisy datum nx_i to the nearest data point of all clusters c_g . So the recovered points will be the member of the idx^{th} cluster such that its minimum pairwise distance is not deviated from the average of pairwise distance in such cluster. **Algorithm PointRecovery** describes the process for data point recovery. Besides, **Algorithm LocalMergingII** uses to merge all small clusters. The small clusters c_{small} are joined together with cluster c_l when the minimum inter-distance $D3_{c_{small}c_l}$ is in the range of the average intra-distance chosen between $D1_{c_{small}}$ and $D1_{c_l}$. Otherwise, two clusters are joined when the center distance between the center of cluster c_{small} and the center of cluster c_l is less than the average center distance of all the clusters without the large clusters. After completing this stage, the small clusters can be merged to the larger clusters. All algorithms used in this stage are orderly shown in the following.

Algorithm PointRecovery

Input: (1) the set of noisy data NX and (2) the set of clusters C .

Output: the set of clusters C with recovered points.

Begin

1. **Repeat**

2. Compute the average intra-distance $D1_{c_g}$ of each cluster c_g using Eq. 3.9,

$$\forall x_i \in c_g.$$

3. Compute the standard deviation intra-distance $D1_{c_g-std}$ of each cluster c_g .

4. **For** each noisy datum nx_i **Do**

5. Compute point inter-distance $D5_{nx_i}$ using Eq. 3.13.

7. Get cluster index idx using Eq. 3.14.

8. **If** $D5_{nx_i}$ is less than $D1_{idx} + (3 \times D1_{idx-std})$ **Then**

9. Recover noisy data nx_i to cluster c_{idx} .
10. **Endif**
11. **Endfor**
12. **Until** there are no noisy data nx_i recovered.

End

Algorithm TemporaryRemoval

Input: the set of clusters C with recovered points.

Output: (1) temporarily removed large clusters and (2) the set of remaining clusters C without the large clusters.

Begin

1. Compute the number of data points n_g of all cluster c_g .
2. Sort all of n_g in descending order.
3. **For** each n_g **Do**
4. Compute the difference $diff_g = n_g - n_{g+1}$.
5. **Endfor**
6. Compute cluster index $idx = arg_g Max_{\forall g; g \neq 1}(diff_g)$.
7. **For** each cluster c_g **Do**
8. **If** n_g is greater or equal to n_{idx} **Then**
9. Temporarily remove cluster c_g .
10. **Endif**
11. **Endfor**

End

Algorithm LocalMergingII

Input: (1) temporarily removed large clusters and (2) the set of remaining clusters C without the large clusters.

Output: the set of new clusters C .

Begin

1. Compute the average cluster center distance $D0_mean$ of all the remaining clusters c_l .
2. **While** all small clusters c_{small} are not merged **Do**
3. Compute intra-distance $D1_{c_{small}}$ using Eq. 3.9, $\forall x_i \in c_{small}$.
4. **For** each cluster c_l **Do**
5. **If** $small \neq l$ **Then**
6. Compute intra-distance $D1_{c_l}$ of cluster c_l using Eq. 3.9, $\forall x_i \in c_l$.
7. Compute inter-distance $D3_{c_{small}c_l}$ using Eq. 3.11, $\exists x_i \in (c_{small} \text{ and } c_l)$.
8. **If** $D3_{c_{small}c_l}$ is less or equal than $Max(D1_{c_{small}}, D1_{c_l})$ **Then**
9. Remove cluster c_{small} to cluster c_l .
10. **Elseif** $D0_{c_{small}c_l}$ is less than $D0_mean$ **Then**
11. Remove cluster c_{small} to cluster c_l .
12. **Endif**
13. **Endif**
14. **Endfor**
15. **Endwhile**
16. Recover temporarily removed large clusters to the set of new clusters C .

End

3.3.4 Refinement Merging

The last step is used for refinement the clusters to obtain the final clusters. The merging process is like in the second step (**Local Merging I**). Therefore, this step selects the cluster that has the highest number of data points as the starting seed cluster. This seed grows to merge the neighboring clusters that have similarity distances. So two clusters are joined together when the maximum inter-distance $D4_{c_g c_i}$ is in the range of the maximum intra-distance between $D2_{c_g}$ and $D2_{c_i}$. This process is shown in **Algorithm RefinementMerging**. Then, this self-growing process successively merges the neighboring subclusters together until the stopping rule is satisfied. After the final clusters are obtained, each the remaining noisy data nx_i and the data points in the noisy subclusters nsc_k are assigned to be the member of the clusters c_g such that their minimum pairwise distances are not deviated from the average of pairwise distance in such clusters. However, there are still some data points which cannot belong to any clusters, since these points are very different from the others. Thus, these points are set as noise. **Algorithm PointAssign** assigns the data points to be the cluster members or noisy data. The algorithms of this stage are described below.

Algorithm RefinementMerging

Input: the set of clusters C .

Output: the set of new clusters C .

Begin

1. Compute intra-distance $D2_{c_g}$ of each cluster c_g using Eq. 3.10, $\forall x_i \in c_g$.
2. **Repeat**
3. Choose cluster c_g which has a highest number of data points as the starting seed.

4. **Repeat**
5. **For** each cluster c_l **Do**
6. **If** $g \neq l$ **Then**
7. Compute inter-distance $D4_{c_g c_l}$ using Eq. 3.12, $\exists x_i \in (c_g \text{ and } c_l)$.
8. **If** $D4_{c_g c_l}$ is less or equal than $[2 \times \text{Max}(D2_{c_g}, D2_{c_l})]$ **Then**
9. Remove cluster c_l to cluster c_g .
10. **Endif**
11. **Endif**
12. **Endfor**
13. Compute new intra-distance $D2_{c_g}$ of cluster c_g using Eq. 3.10, $\forall x_i \in c_g$.
14. **Until** there are no clusters c_l merged to cluster c_g .
15. **Until** there are no clusters c_g merged.

End

Algorithm PointAssign

Input: (1) the set of clusters C , (2) the set of noisy data NX , and (3) the set of noisy subclusters NSC .

Output: (1) the set of final clusters C and (2) the set of remaining noisy data NX .

Begin

1. Transfer the data point in the noisy subclusters nsc_k to the noisy data nx_i .
2. **Repeat**
3. Compute the intra-distance $D2_{c_g}$ of each cluster c_g using Eq. 3.10, $\forall x_i \in c_g$.
4. **For** each noisy datum nx_i **Do**
7. Compute point inter-distance $D5_{nx_i}$ using Eq. 3.13.
8. Get cluster index idx using Eq. 3.14.
9. **If** $D5_{nx_i}$ is less than $D2_{idx}$ **Then**

10. Assign the noisy datum nx_i to cluster c_{idx} .
11. **Endif**
12. **Endfor**
13. **Until** there are no noisy data nx_i assigned .

End

3.4 Clustering Example

Fig. 3.13 shows an example of how the input data points are delineated. Fig. 3.13(a) is the bivariate input data set of three clusters. In Phase 1, the SPSM starts with partitioning the input data set to automatically obtain the set of subclusters based on the DTS-SOM scheme as shown in 3.13(b) and Fig. 3.13(c). From this result, the DTS-SOM partition this example into 100 subclusters. All of these subclusters are performed noise removal process in Phase 2 which produces the set of noisy data, the set of noisy subclusters, and the set of dense subclusters. Fig. 3.13(d) shows the set of 50 dense subclusters.

The SPSM algorithm performs the sequential merging steps. First, some data points must be recovered from the set of noisy data, since such recovered data points can help the further merging process to satisfy the merging conditions. Next, this step starts to merge between subclusters and their neighboring subclusters which are close to each other based on the threshold value. Therefore, two subclusters are merged when the minimum pairwise distance between subclusters is less or equal than the maximum pairwise distance within subclusters. The **Neighboring Merging** stage results in 25 clusters as shown in Fig. 3.13(e). From this result, there exists some adjacent clusters can be joined together to the larger clusters, since there may be no the neighborhood connections among those adjacent clusters. For example, in Fig. 3.13(e) considering the

cluster c_1 and c_4 denoted by the numbers 1 and 4, these two clusters are adjacent clusters but they cannot be merged by the **Neighboring Merging** step. When considering the distances, the minimum pairwise distance between clusters is in the range of the average pairwise distance within clusters. In **Local Merging I**, these two adjacent clusters can be merged to the same cluster denoted by c_1 as shown in 3.13(f). In the same way, the cluster c_3 and c_{16} denoted by the numbers 3 and 16 as shown in 3.13(e) are also merged to form the larger clusters denoted by c_3 as shown in 3.13(f).

Before performing the further step, some data points are recovered from the set of noisy data as depicted in Fig. 3.13(g). Fig. 3.13(h) shows the small clusters and the remaining clusters after temporary removal the large clusters based on the threshold value. So in this figure, the cluster c_1 , c_3 , and c_{10} are temporarily discarded. After that, all possible small clusters are joined together to the larger clusters. Fig. 3.13(i) shows the merging result after performing the **Local Merging II** process with recovering the large clusters. In the **Refinement Merging** step, all the clusters are merged together if the maximum pairwise distance between the clusters is less or equal to the twice of the maximum pairwise distance within clusters. Then, each data point in the set of noisy data and the data points in the set of noisy subclusters are assigned to be the member of the cluster. The property of the recovered points are not deviated from the other data points in such clusters. Otherwise, the deviated data points are assigned as noise. Fig. 3.13(j) illustrates the final clustering result. From this result, this example is extracted into three clusters which are automatically obtained from the SPSM method.

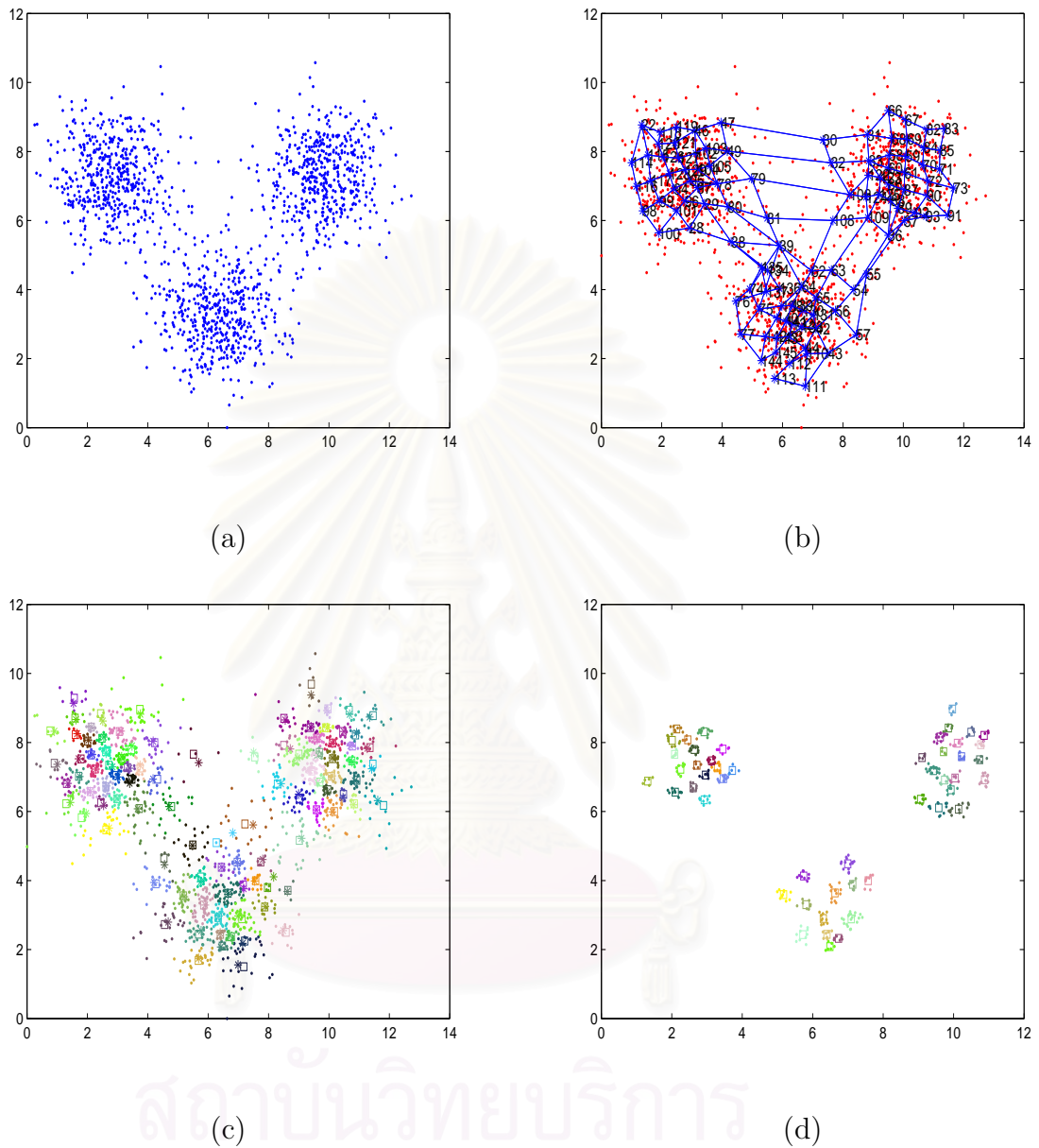
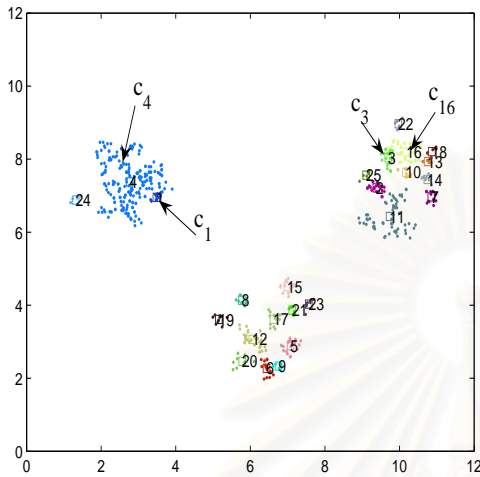
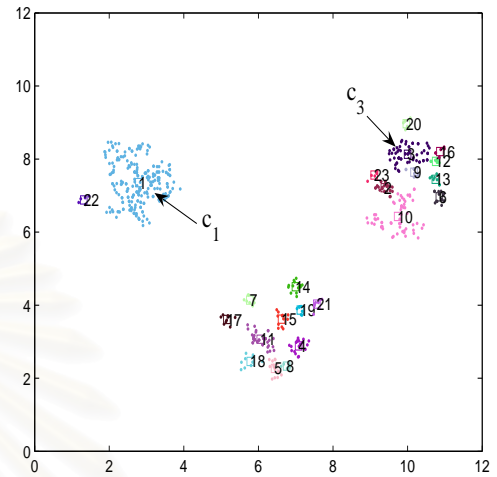


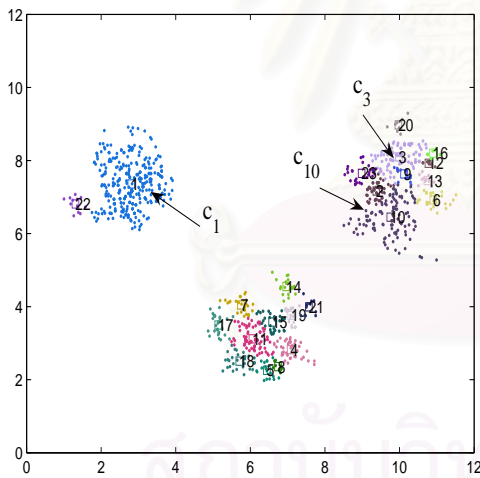
Figure 3.13: An example how the SPSM works. (a) Input data set of three clusters. (b) The lateral connections among the leaf nodes. A number shows the node index. (c) The set of 100 subclusters obtained through Phase 1. (d) The set of 50 dense subclusters. Each subcluster is denoted by a color. Each star is represented a prototype vector obtained from the **DTS-SOM Training** process and each square is represented a subcluster center.



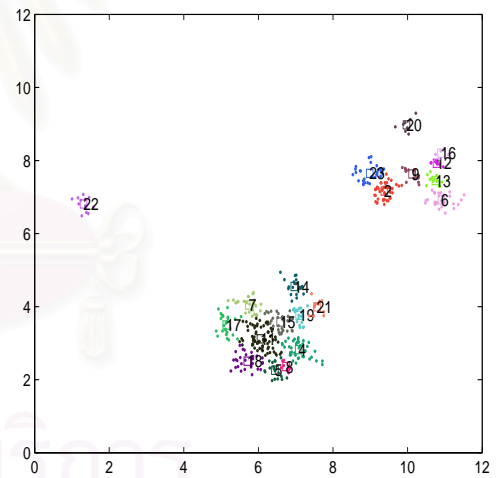
(e)



(f)

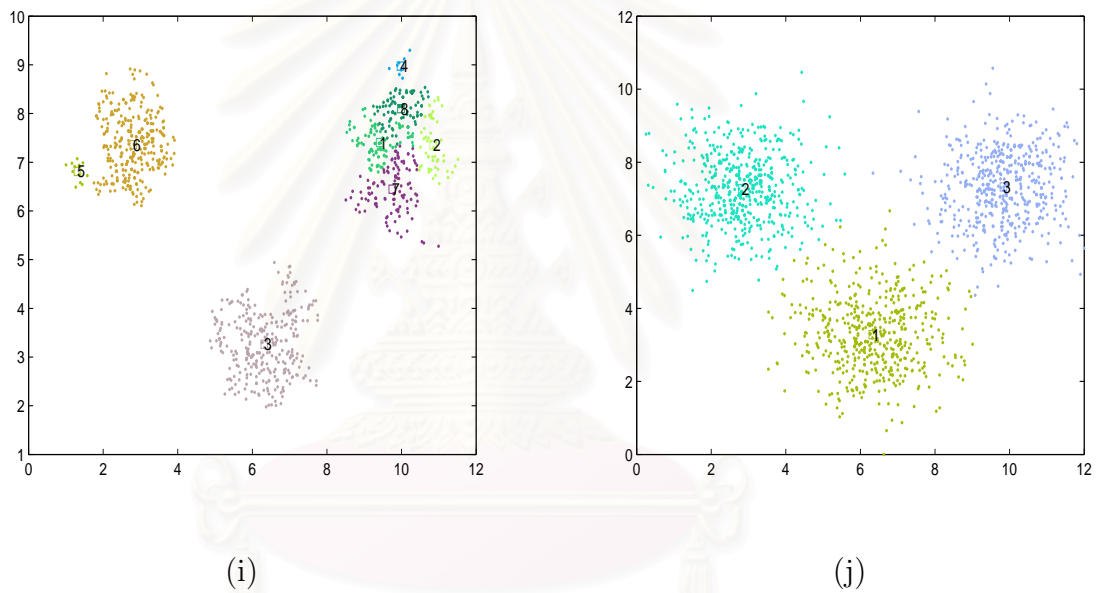


(g)



(h)

(e) The merging result obtained from the **Neighboring Merging** step (25 clusters). (f) The cluster merging result obtained through the **Local merging I** step. (g) The data point filling result acquired from the **Algorithm PointRecovery**. (h) The temporarily removed cluster result.



(i) The cluster merging result obtained through the **Local Merging II** step. (j) The final clustering result (3 clusters). Each cluster is denoted by a color and a number.

จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER IV

EXPERIMENTAL RESULTS

The SPSM algorithm has been implemented and tested for correctness and performance by MATLAB version 7.0. All tests were performed on Sony laptop with 1.6 MHz Pentium IV processor and 1 GB of RAM, running on Microsoft Windows XP. The SPSM algorithm makes use of the following parameters especially in the Phase 1 as depicted in Table 4.1. Since the initialization of the DTS-SOM training starts at the center of the data and all the data set used in our experiments has the large number of data points which are more than one thousand data points, the activation level should not set too large. If the activation level is set too large, this will affect the network taking a long time for learning and being disorderly. Thus, the activation level τ_i was chosen to 0.05 of the number of input data. At each spitting stage set $\Delta\tau = 0.05\tau_i$ for all node i .

Besides, the learning rate α should begin with a small value. That means the learning rate α will set close to 0.1. The width of neighborhood function σ should initially include the neighboring neurons centered on the best matching unit. So these experiments will set to 0.99. SPSM algorithm decides to stop the DTS-SOM training when the difference of the size of tree nodes between the current and the previous epochs is less than 0.005. All the experiment results described here were obtained with these parameters. The details of each data set used in this dissertation as shown in Table 4.2. For Data Set 6 to Data Set 8 are reported in [12]. The experimental results are shown in the following sections.

Table 4.1: Phase 1 parameters.

Parameter	Ranges
The activation level τ_i	$0.05N-0.5N$
Splitting rate $\Delta\tau$	$(0, 1]$ of the activation level τ_i
The learning rate α	< 1
The width of neighborhood function σ	$0.8-0.99$

4.1 Experiment 1

The first experiment has tested on data set as shown in Fig. 4.1(a). Data Set 1 is generated from three normal distributions with covariance I , where I is 2×2 identity matrix and mean vectors $(3, 7)$, $(10, 7)$, and $(6.5, 3)$, respectively. The purpose of this experiment is to test the performance of the SPSM algorithm which can decompose the non-linearly separable clusters. After performing SPSM algorithm, 133 nodes are produced from the DTS-SOM training and the three extracted clusters are depicted in Fig. 4.1(b).

4.2 Experiment 2

This experiment performed on the data set as shown in Fig. 4.2 and Fig. 4.3. All data sets are generated from four normal distributions with covariance I . The mean vectors of Data Set 2 are $(3.55, 8.25)$, $(8.5, 8.25)$, $(3.55, 3)$, and $(8.5, 3)$ as depicted in Fig. 4.2(a). For Data Set 3, mean vectors are $(3.25, 7)$, $(7.25, 7)$, $(3.25, 3)$, and $(7.25, 3)$ as shown in Fig 4.2(c). For 4.3(a), the mean vectors of Data Set 4 are $(3.5, 7.75)$, $(7.75, 7.75)$, $(3.5, 3)$, and $(7.75, 3)$. The clusters in Data Set 3 are closer than the clusters in Data Set 2. Besides, more data points are added and very close to each other in Data Set

Table 4.2: Details on each data set.

	The number of input data (N)	The desired number of clusters (K)
Data Set 1	1500	3
Data Set 2	2400	4
Data Set 3	2400	4
Data Set 4	4800	4
Data Set 5	2800	3
Data Set 6	8000	6
Data Set 7	10000	9
Data Set 8	8000	8
Data Set 9	2689	4

4. These data sets are used to test tolerance to noise of the SPSM algorithm. After completing the DTS-SOM training, the tree nodes are produced into 153, 277 and 241 nodes for Data Set 2, Data Set3 and Data Set 4, respectively. As shown in Figs. 4.2(b), 4.2(d), and 4.3(b), the SPSM algorithm is able to successfully partition these data sets. Moreover, the number of clusters obtained from the SPSM algorithm as depicted in the parenthesis, is the same as the desired number of clusters as shown in Table. 4.2.

4.3 Experiment 3

This experiment is used to compare with other clustering algorithms. The data sets used in this experiment are obtained from [12] which consisted of Data Set 6, Data Set 7, and Data Set 8. The DTS-SOM training gives the size of tree nodes into 369, 305, and 489 for Data Set 6, Data Set 7, and Data Set 8, respectively. The clustering results of

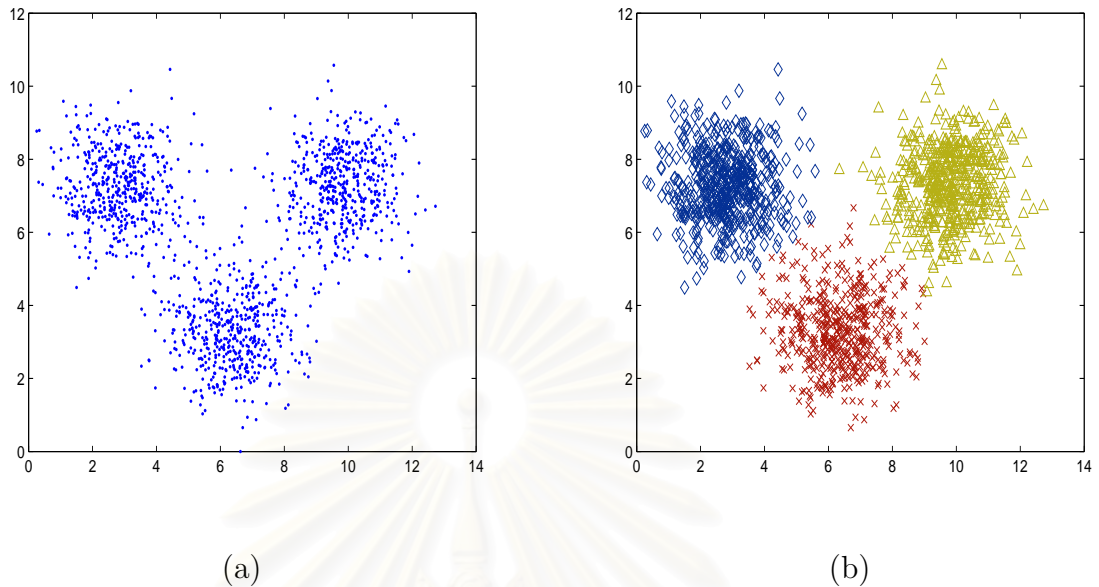
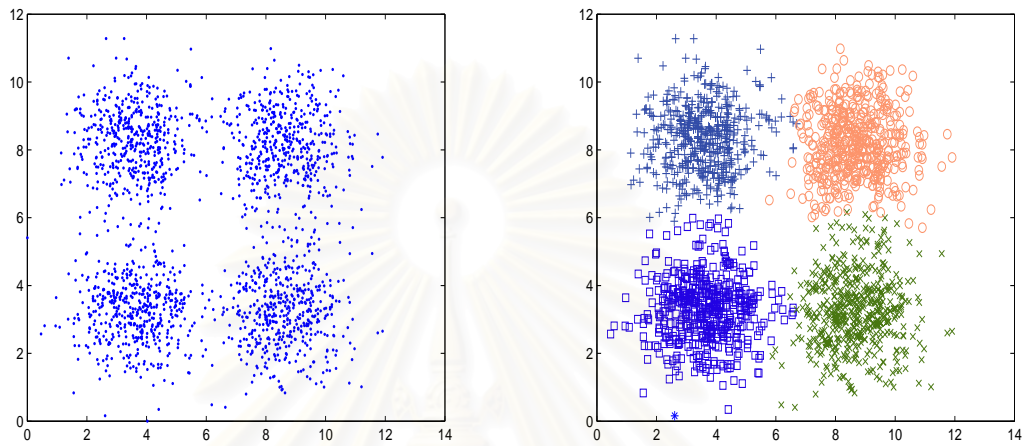


Figure 4.1: Clustering result on the experiment 1. (a) Data Set 1. (b) SPSM on Data Set 1 (3 clusters). Each cluster is denoted by a symbol.

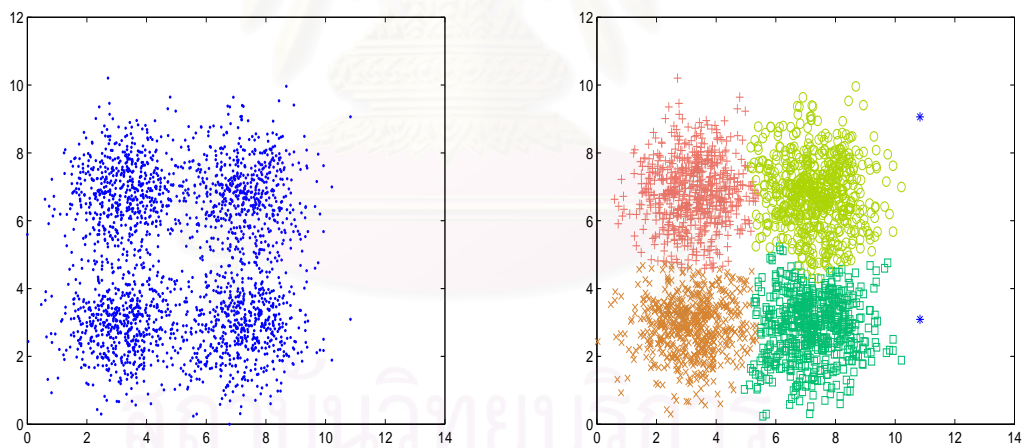
SPSM algorithm are shown in Fig. 4.4 and Fig. 4.5. From the results, SPSM algorithm can obtain the correct clustering results and can identify the noisy data. The number of clusters produced by SPSM algorithm as depicted in the parenthesis is the same as the desired number of clusters as shown in Table. 4.2. These data sets have been tested with the other algorithms. The single-link and complete-link algorithms are implemented by using the Statistics Toolbox in Matlab version 7.0. The program of algorithm CURE is obtained from public domain at <http://www.cs.cas.cz/~petra/Liter-Odkazy-shluk.html>.

Some of unsuccessful clustering results acquired from the single-link and complete-link algorithms are shown in Fig. 4.6, because these clustering algorithms are sensitive to noises. Besides, the output of these two clustering algorithms is a hierarchical tree which can be cut at a desired level forming a clustering result. If this tree is cut at different levels, the different clustering results can be obtained. The level has been chosen to obtain the desired number of clusters as depicted in the parenthesis in Fig.



(a) Data Set 2.

(b) SPSM on Data Set 2 (4 clusters).



(c) Data Set 3.

(d) SPSM on Data Set 3 (4 clusters).

Figure 4.2: Clustering results on the experiment 2 for Data Set 2 and Data Set 3. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.

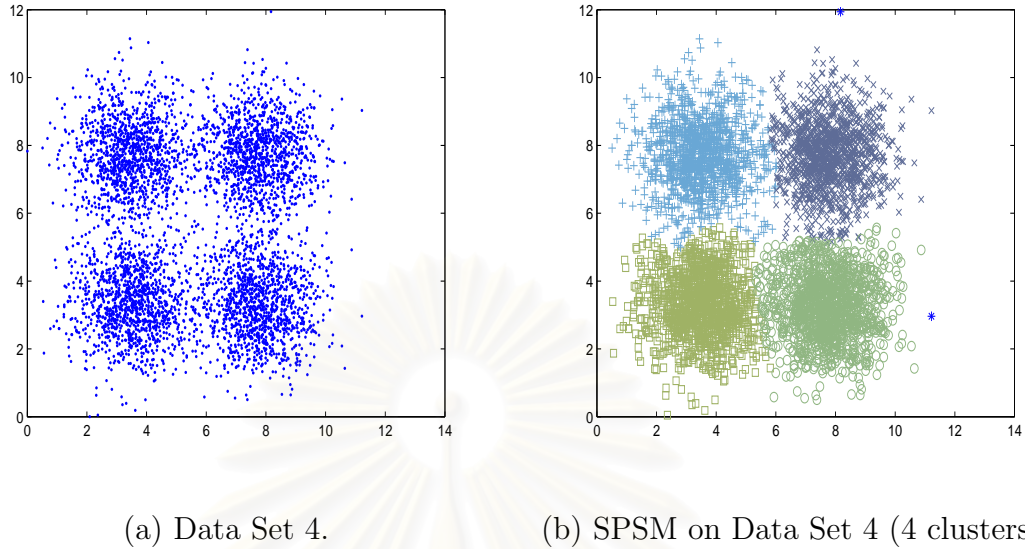
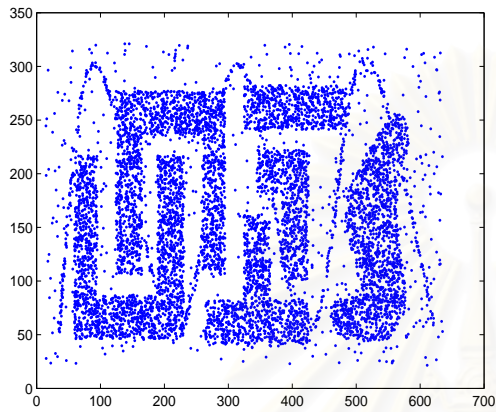


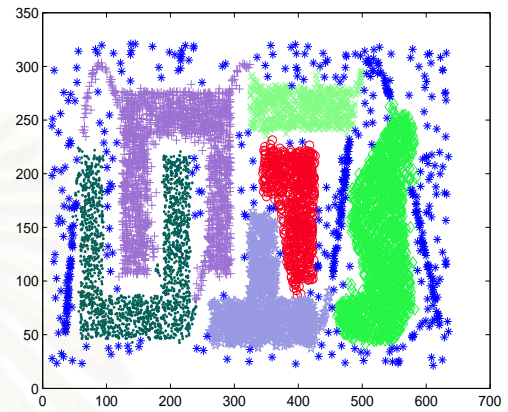
Figure 4.3: Clustering results on the experiment 2 for Data Set 4. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.

4.6(a)-(d). As well as CURE algorithm, CURE fails on these data sets as illustrated in Fig. 4.7. Recall, CURE algorithm shrinks the representatives for each cluster toward the center of cluster in order to eliminate the effects of noise. However, the shrinking method may cause some clusters to be split. On the other hand, if the shrinking method is weakened, some clusters will be merged by noise links. So the shrinking parameter must carefully choose. Fig. 4.8 shows the results of algorithm CSM acquired from <http://arbor.ee.ntu.tw/~owenlin/tkde.csm/>. The CSM algorithm may fail to partition the input data set because of the parameter selection. One of the parameters may affect the correctness of the partition is that the parameter m (the number of subclusters) specified by users. From the results, the subclusters produced in phase one may be too many. So it may cause that the subclusters which should be noisy subclusters will become the dense subclusters. This will affect the correctness of the further subsequent merging. On the other hand, when the value of parameter m is too small that means the number of subclusters obtained from phase one is too low. So many noisy subclusters

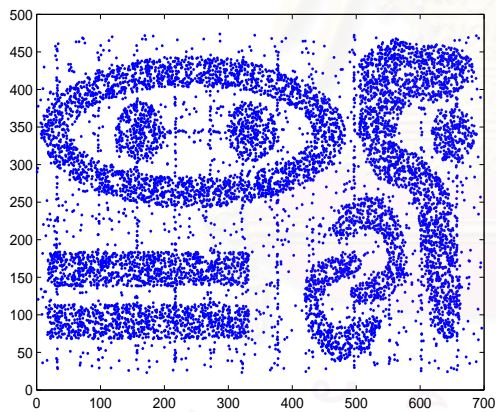
may be existent. These noisy subclusters may merge with other clusters and affect the clustering results. Like CURE algorithm, the parameters of CSM algorithm must be carefully chosen.



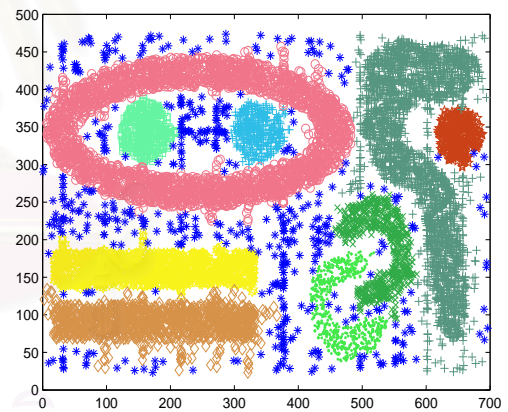
(a) Data Set 6.



(b) SPSM on Data Set 6 (6 clusters).

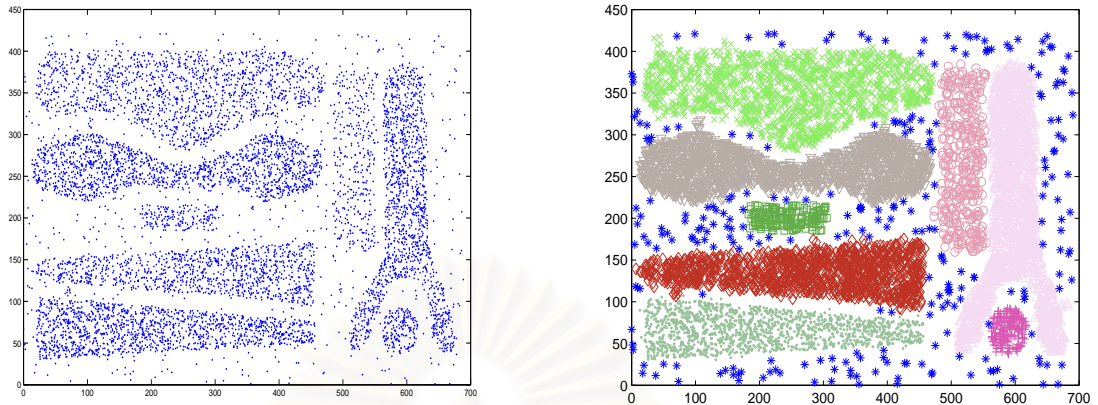


(c) Data Set 7.



(d) SPSM on Data Set 7 (9 clusters).

Figure 4.4: Clustering results on the experiment 3 produced in SPSM algorithm. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.



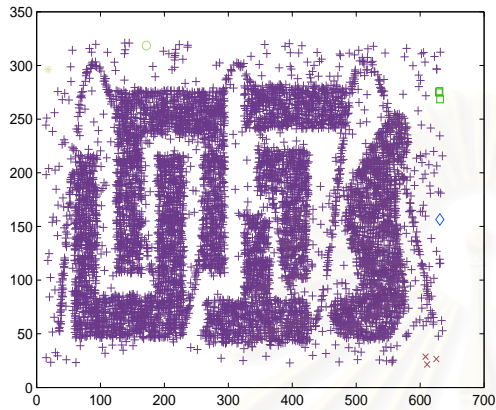
(a) Data Set 8.

(b) SPSM on Data Set 8 (8 clusters).

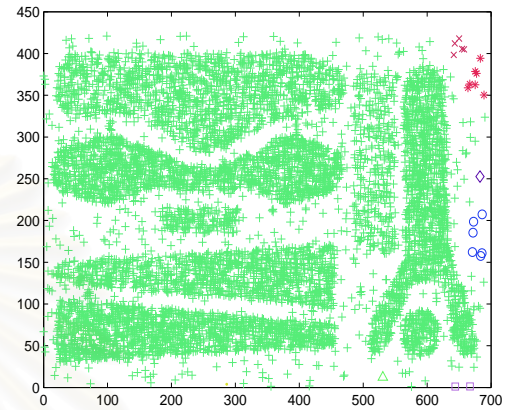
Figure 4.5: Clustering results on the experiment 3 produced in SPSM algorithm. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.

4.4 Experiment 4

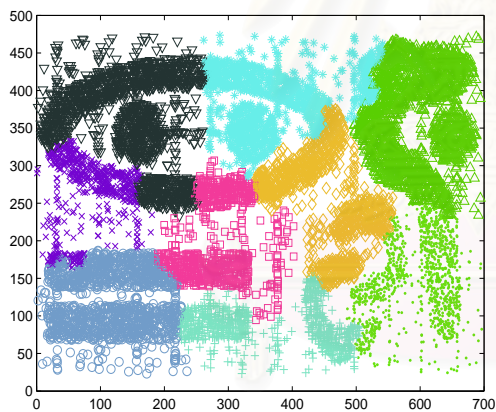
In Data Set 5 as depicted in Fig. 4.9(a), there is one large cluster and two small clusters. This data set is generated from three normal distributions with two different covariances D_1 and D_2 . D_1 is the square diagonal matrix in which the diagonal entries are 6 and 3.5 for the large one. D_2 is also the square diagonal matrix in which both diagonal entries are 2 for the others. The mean vectors are $(7, 18)$, $(7, 8)$ and $(20, 12)$. The purpose of this experiment is to test the ability to detect different volumes of clusters. The number of tree nodes is equal to 169 nodes when completing the DTS-SOM training. Fig. 4.9(b) illustrates this ability which can produce the three identified clusters as depicted in the parenthesis.



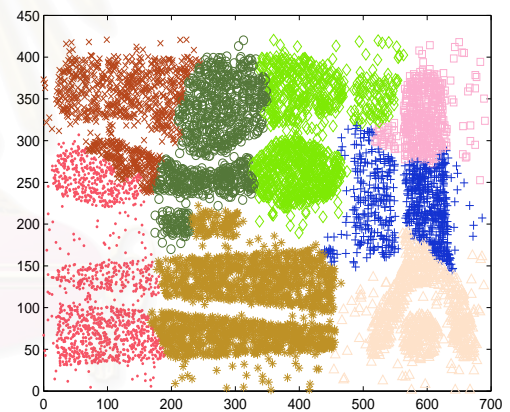
(a) Data Set 6 (6 clusters).



(b) Data Set 8 (8 clusters).



(c) Data Set 7 (9 clusters).



(d) Data Set 8 (8 clusters).

Figure 4.6: Clustering results on the experiment 3. (a) and (b) are the final decomposition of Data Set 6 and Data Set 8 obtained from the single-link algorithm, respectively. (c) and (d) are the clustering results of Data Set 7 and Data Set 8 acquired from the complete-link algorithm, respectively. Each cluster is denoted by a symbol.

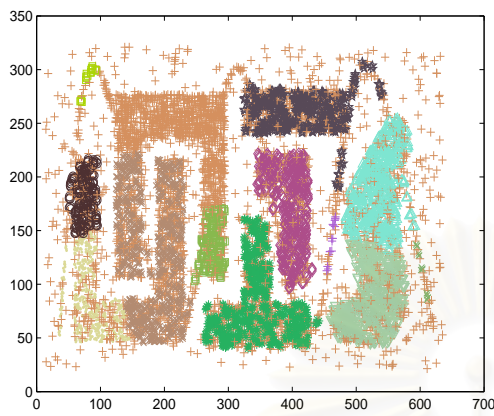
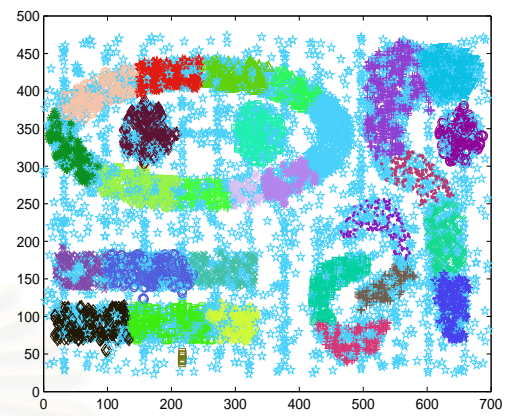
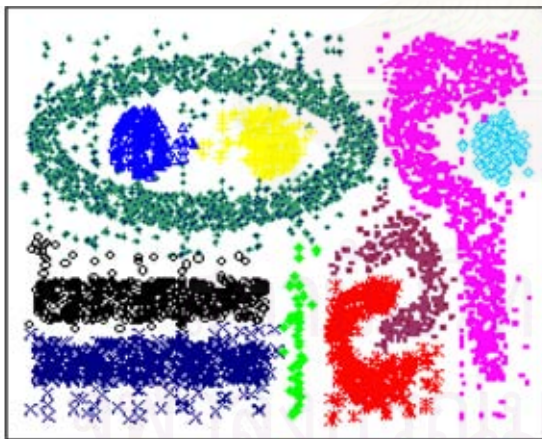
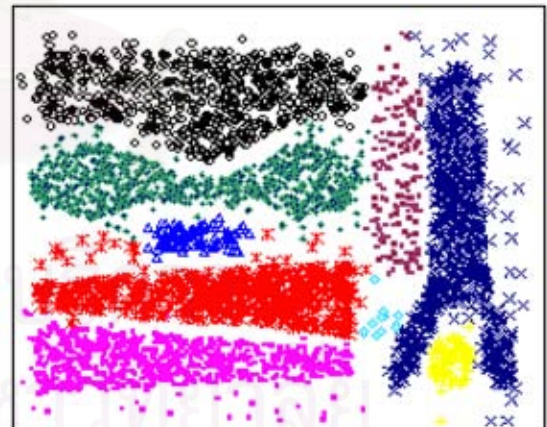
(a) Data Set 6 (> 6 clusters).(b) Data Set 7 (> 9 clusters).

Figure 4.7: Clustering results on the experiment 3 obtained through the algorithm CURE. (a) and (b) are the final decomposition of Data Set 6 and Data Set 7, respectively. Each cluster is denoted by a symbol.



(a) Data Set 7 (10 clusters).



(b) Data Set 8 (9 clusters).

Figure 4.8: Clustering results on the experiment 3 produced by the algorithm CSM. (a) and (b) are the final decomposition of Data Set 7 and Data Set 8, respectively. Each cluster is denoted by a symbol.

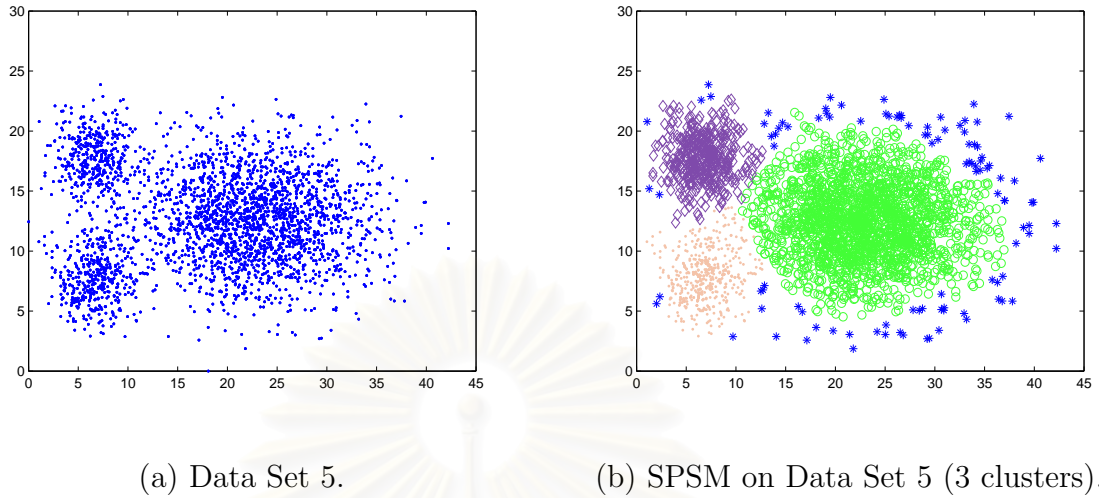


Figure 4.9: Clustering result on the experiment 4 for Data Set 5. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.

4.5 Experiment 5

The purpose of this experiment is to test the ability to detect different density of clusters. Data Set 9 is generated from four normal distribution with three different density values as shown in Fig. 4.10(a). The mean vectors are $(6, 10)$, $(12, 12)$, $(5, 5)$ and $(11, 6)$. After performing the DTS-SOM training, 149 tree nodes are produced. Fig. 4.10(b) illustrates this ability which can produce the four identified clusters as depicted in the parenthesis. But the algorithms CURE and CSM fail to find the right clusters. Because the shrinking method in the CURE algorithm may cause some clusters to be vanished especially the sparse clusters. Besides, the merging process of the CSM algorithm will merge any two subclusters together which have the highest density to form the larger clusters. The CSM algorithm assumed that the number of points in a noisy cluster is much less than that of a normal cluster. It is possible that the clusters which have the less density may be considered to be the noisy clusters and such clusters will be removed eventually.

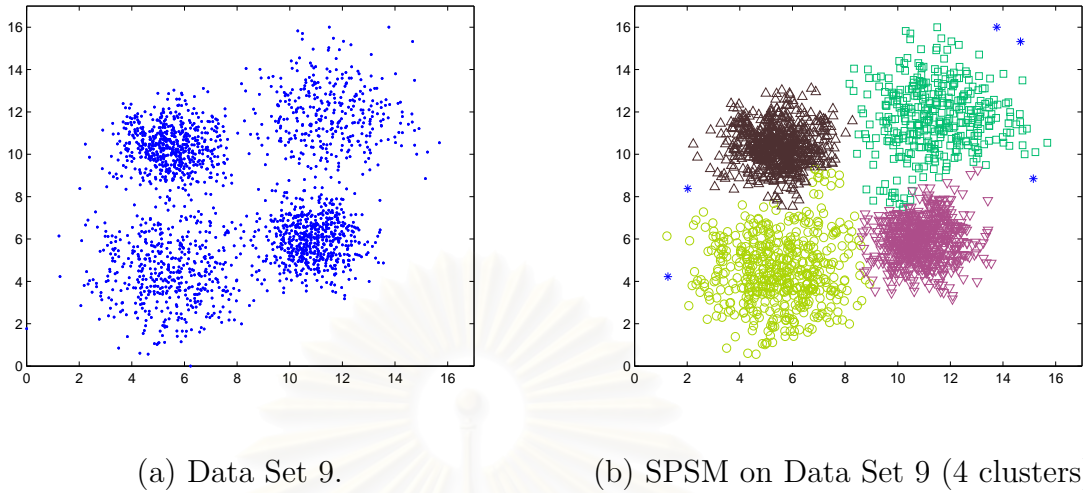


Figure 4.10: Clustering result on the experiment 5 for Data Set 9. Each cluster is denoted by a symbol. The star symbol is represented noisy data points.

4.6 Experiment 6

Finally, this experiment is used to test a sensitivity analysis on the parameter settings especially the threshold value of stopping criterion in DTS-SOM training. With different values of stopping criterion, the different initial subclusters can be obtained. Fig. 4.11(a) shows the clustering results when using the stopping criterion as 0.005. The number of initial subclusters is 221 subclusters. Fig. 4.11(b) shows the results when using the stopping criterion as 0.0025. The DTS-SOM training produces 239 subclusters. From these results, the algorithm SPSM is still able to extract to the correctness results which are 9 clusters, and SPSM algorithm can identify the noisy points.

In addition, another sensitivity analysis on the parameter settings is the initialization of the first neuron in the DTS-SOM training. The previous experiments start by taking a single neuron and placing it at the center of mass of the data cloud. Fig. 4.12 shows the clustering results when selecting the prototype vector of the first neuron from the available set of input vectors in a random manner. From the results, the algorithm

SPSM is able to extract to the right clusters.

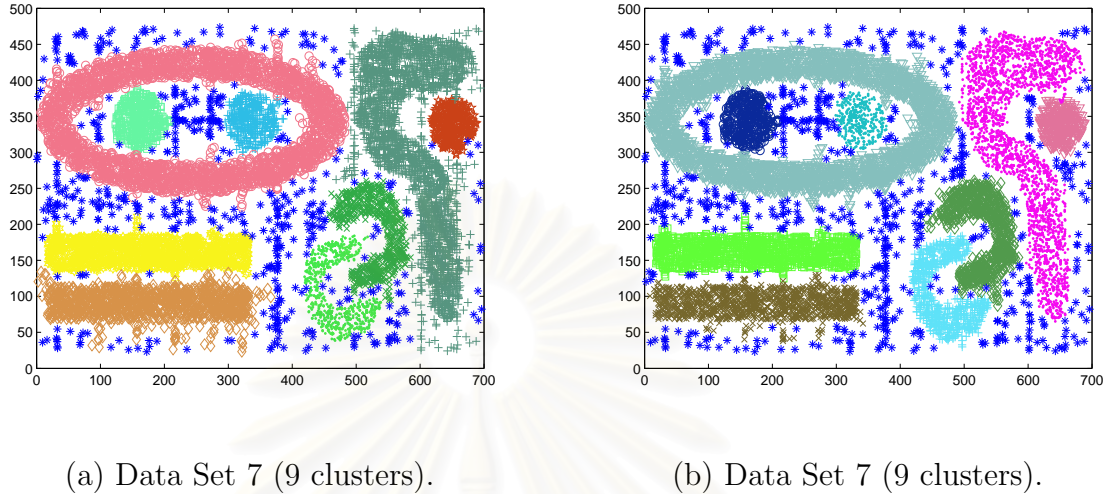


Figure 4.11: Clustering results on the experiment 6 obtained through the algorithm SPSM. (a) and (b) are the final decompositions of Data Set 7 with different stopping criteria. Each cluster is denoted by a symbol.

4.7 Experiment 7

This experiment is used to test the robustness to noisy data points of the SPSM algorithm according to the objective of this dissertation. The data sets used in this experiment are generated in arbitrary shape with only one clusters consisted of 1153 data points. In addition to the clustered data points, noise in form of data points uniformly distribution throughout the overview of the data sets are added to the data sets. The parameter p controls the percentage of data points in the data set that are considered noise. This experiment has tested the sensitivity of the parameter p by using $p = \{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$. Five data sets are generated for each parameters p . Fig. 4.13 shows one of the data set with 5% of noise. Table 4.3 depicts the average of the accuracy of the clustering results obtained from the SPSM algorithm. The accuracy

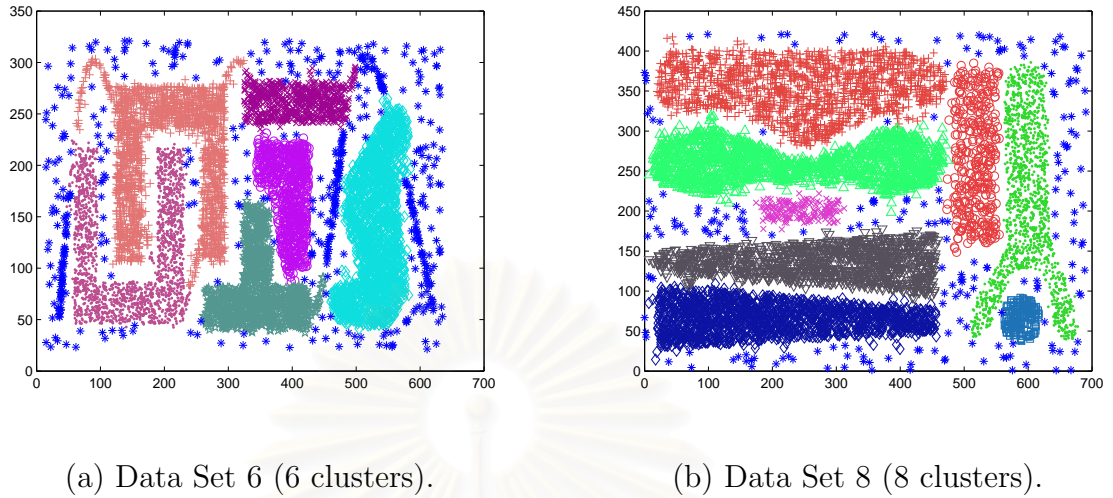


Figure 4.12: Clustering results on the experiment 6 obtained through the algorithm SPSM. (a) and (b) are the final decompositions of Data Set 6 and Data Set 8 with different initialization of the first neuron. Each cluster is denoted by a symbol.

of the clustering result is measure by using Eq. 4.1,

$$accuracy (\%) = \frac{N - MP}{N} \times 100 \quad (4.1)$$

where N is the number of data point and MP is the number of miss-classified data points. As shown in Table 4.3, the SPSM algorithm gives the high accuracy of the clustering results when the percentage of noisy data is lower, and the SPSM algorithm can be robust to noisy data up to 30%. The correctness of clustering results depends on the percentage of noisy data and the closeness of noisy data to the actual data points. When the noisy data are more closer to the actual data points, the accuracy of the clustering results is more less. However, from Table 4.3 can conclude that the SPSM algorithm can tolerate to the noisy data.

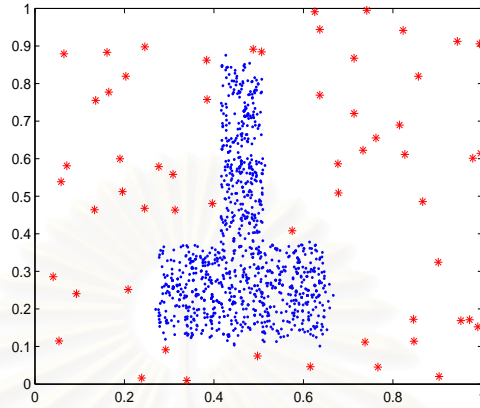


Figure 4.13: An example of data set used in the experiment 7. The dots represented the data points and the stars denoted the noisy data points

Table 4.3: The average of the accuracy of clustering results obtained through the SPSM algorithm for testing the tolerance to noisy data.

Percentage of noisy data	The average of the accuracy of clustering results (%)
5 %	99.92 \pm 0.0850
10 %	99.78 \pm 0.0876
15 %	99.53 \pm 0.1354
20 %	99.51 \pm 0.0798
25 %	99.38 \pm 0.0585
30 %	99.19 \pm 0.1688

4.8 Complexity Analysis

Since, the SPSM algorithm consists of three phases, the computational complexity is the sum of three phases. The computational complexity is described in the following.

1. Complexity of Phase 1.

In Phase 1, the SPSM algorithm applies the DTS-SOM on the set of N input data points to obtain M subclusters. So the computational complexity is the sum of finding the best matching unit C_{BMU} , updating the neighbors C_{UDH} , and updating the other internal nodes C_{UDP} .

To find the best matching unit, a vector distance is calculated to b times at every level of the tree. If h is the height of the tree, the amount of calculations needed is

$$C_{BMU} = h \cdot b \quad (4.2)$$

Updating the neighboring nodes, k nodes are updated. Besides, the leaf nodes are split when their counters become to zero. The splitting of the leaf nodes happens on average every $\frac{1}{\beta}$ step.

To update the parent nodes (internal nodes), the parent nodes will only be updated if their children are also updated. This updating is proceeded layer by layer back to the root node. So the time calculations is the height of the tree. The amount of calculations needed for a epoch size N is

$$C_{tot} = N(C_{BMU} + C_{UDH} + C_{UDP}) \quad (4.3)$$

$$= N\left(h \cdot b + \left(k + \frac{1}{\beta}\right) + h\right) \quad (4.4)$$

Note that b is the constant. Since each internal node has four children, so this variable can be ignored. In during training process, a 4-branch search tree is

formed. So h is changed according to $\log_4 N$. For the amount of updated neighbors k , at least eight direct neighbor nodes are updated. Then k can approximate to a constant time and β is a constant as well. The computational complexity of Phase 1 can reduce to $O(N \cdot \log_4 N)$.

2. Complexity of Phase 2.

For Phase 2, the SPSM algorithm will discard the noisy data including the noisy subclusters from the set of subclusters produced by Phase 1. This process composes of multiple orderly stages. However, the complexity calculation only depends on the first step (**Algorithm DensityFinding**) that discards the sparse data points and computes the density values for each subcluster. In this process, every data points for each subcluster are examined. The amount of calculations needed for every subclusters is

$$C_{tot} = \sum_{j=1}^M |sc_j| \quad (4.5)$$

where, M is the number of subclusters such that $M < N$. Therefore, the time complexity of this process is linear to the size of subclusters, that is $O(N)$. The computational complexity of Phase 2 is also $O(N)$.

3. Complexity of Phase 3.

In Phase 3, algorithm SPSM adapted the agglomerative clustering which is self-merging process. This phase also consists of multiple orderly stages. The most time complexity is in the **Refinement Merging** process. Since some data points are recovered from the set of noisy data, it takes more times than the other steps for the intra-distance computation. Then the computational complexity is the sum of intra-distance computation of each cluster and self-merging process. If each cluster c_j consists of a set of n_0 data points, the amount of intra-distance calculation for

any clusters c_j is

$$C_{intra} = \sum_{\forall i} \sum_{\forall k; i \neq k} \|x_i - x_k\| = n_0^2 \quad (4.6)$$

The amount of calculations needed for G clusters is $G \cdot n_0^2$ such that $G < n_0 < N$. Then, the computational complexity of intra-distance calculations for G clusters that approximates to $O(N^2)$.

During each self-merging process, some data points of two clusters are used for computing the inter-distance. So the time computation for inter-distance calculation is less than that for intra-distance calculation which approximates to $O(N^2)$. Then, the computational complexity of Phase 3 is $O(N^2 + N^2) = O(N^2)$.

Thus, the overall complexity of the SPSM algorithm is $O(Phase1 + Phase2 + Phase3) = O(N \cdot \log_4 N + N + N^2) \approx O(N^2)$.

Table 4.4 shows the time complexity of the SPSM algorithm including the other algorithms. Note that parameter m for the CSM algorithm is the number of subclusters. The SPSM algorithm requires more computational time than the CSM algorithm. This is because the SPSM algorithm takes time to compute the distance criteria. When compared with the rest algorithms, the time complexity of algorithm SPSM is better

Table 4.4: Summary of computational complexity of different algorithms.

Clustering Algorithm	Complexity
Single-link	$O(N^2 \log N)$
Complete-link	$O(N^2 \log N)$
CURE	$O(N_{sample}^2 \log N_{sample})$
SPSM	$O(N^2)$
CSM	$O(mN + m^2 \log m)$

than that of the rest algorithms. However, from the results, it demonstrated that the proposed algorithm SPSM is very efficient clustering which is able to the handle noisy and arbitrary shapes data set.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER V

CONCLUSION

Data clustering algorithm attempts to organize unlabeled input data into clusters or natural groups within a cluster are more similar to each other than data points belonged to different clusters. The existing clustering algorithms, such as single-link clustering, complete-link clustering, k-means, CURE, and CSM are designed to find clusters based on pre-defined parameters. These algorithms fail if the choice of parameters is incorrect with respect to the data set being clustered. Most of these algorithms work very well for compact and hyperspherical clusters.

In this dissertation, the new hybrid clustering called Self-Partition and Self-Merging (SPSM) is proposed. The SPSM algorithm has been designed into three phases. In Phase 1, the new partitioning clustering is introduced based on a self-creating and self-organizing algorithm designed to improve SOM algorithm called Dynamic Tree-Structured Self-Organizing map (DTS-SOM). After performing the DTS-SOM training, the number of initial subclusters is automatically obtained. To achieve a better clustering result and be less affected by noises, the noisy data and the noisy subclusters are removed by Phase 2. Then, the algorithm SPSM performs self-merging process in Phase 3 based on inter-distance and intra-distance criteria. The SPSM algorithm automatically obtains the final clusters and can identify the noisy data.

The time complexity of the SPSM algorithm is $O(N^2)$, where N is the number of data points, as described in Chapter 4. The SPSM algorithm requires more computational time than the CSM algorithm. This is because the SPSM algorithm takes time to compute the distance criteria. When compared with the other algorithms, the time

complexity of algorithm SPSM is better than that of the other algorithms.

From the experimental results, algorithm SPSM is able to cluster the data sets of arbitrary shapes very efficiently, tolerate to noise, and provide better clustering results than the existing clustering algorithms. The main contributions of our proposed method can be summarized as follows:

- The Dynamic Tree-Structured Self-Organizing map (DTS-SOM) is proposed to cope with the initialization of the number of clusters required in the partitioned clustering algorithm. The DTS-SOM is a variant of SOM which is a self-creating and self-organizing algorithm designed to improve SOM algorithm. Using DTS-SOM, it is able to overcome the limitations of SOM. Since the SOM must pre-define the topology structure and the number of neurons before the training process. Moreover, the correctness of finding the best matching unit using the DTS-SOM training is more than the correctness of finding the best matching unit using the ETree algorithm.
- The SPSM algorithm also proposes the noise removal method which can deal with the noisy data set. As the results of all experiments and Table 4.3 used to test the noise robustness, the algorithm SPSM is able to not only resist noises, but also lead to good clustering results. Unlike the other algorithms, the SPSM algorithm can identify which data points is the noisy data.
- The SPSM algorithm is able to cluster the data sets of arbitrary shapes very efficient and provide better results than the other algorithms. Moreover, the SPSM algorithm is also capable to detect different density of clusters as shown in Fig. 4.10.
- Our proposed method's requirements are minimum to determine input parameters. The parameter requirements are the common parameters which only used for the

DTS-SOM training. From the experiment for testing a sensitivity analysis, SPSM algorithm is still able to extract the correctness number of clusters.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

References

1. Jain, A.K.; Murty, M.N.; and Flynn, P.J. Data clustering: A review. ACM Computing Surveys. 31:3(1999): 264 - 323.
2. Ankerst, M.; Breunig, M.M.; Kriegel, H.P. and Sander, J. OPTICS: Ordering points to identify the clustering structure. Proc. ACM SIGMOD Int. Conf. on Management of Data. pp. 49 - 60. Philadelphia, Pennsylvania, 1999.
3. Duda, R.O.; Hart, P.E.; and Stork, D.G. Pattern Classification. second ed. Wiley, 2001.
4. Kaufman, L. and Rousseeuw, P.J. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, 1990.
5. Berkhin, P. Survey of clustering data mining techniques [Online]. 2001. Available from: http://www.accrue.com/products/rp_clustering_review.pdf
6. Kolatch, E. Clustering algorithms for spatial databases: A survey [Online]. 2001. Available from: <http://citeseer.nj.nec.com/436843.html>
7. Xu, R. and Wunsch, D. Survey of clustering algorithms. IEEE Trans. Neural Networks. 16:3(2005): 645-678.
8. Kanungo, T.; Mount, D.; Netanyahu, N.; Piatko, C.; Silverman, R.; and Wu, A. An efficient K-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. 24:7(2000): 881-892.
9. Su, M. and Chou, C. A modified version of the K-means algorithm with a distance based on cluster symmetry. IEEE Trans. Pattern Anal. Mach. Intell. 23:6(2001): 674-680.
10. Guha, S.; Rastogi, R.; and Shim, K. CURE: An efficient clustering algorithm for

- large databases. Proc. ACM SIGMOD Int. Conf. on Management of Data. 1998: 73 - 84.
11. Guha, S.; Rastogi, R.; and Shim, K. ROCK: A robust clustering algorithm for categorical attributes. Inf. Syst. 25:5(2000): 345-366.
 12. Karypis, G.; Han, E.; and Kumar, V. Chameleon: A hierarchical clustering using dynamic modeling. IEEE Computer. 32:8(1999): 68-75.
 13. Hertz, J.; Krogh, A.; and Palmer, R.G. Introduction to the Theory of Neural Computation. Reading, Mass.:Addison-Wesley, 1990.
 14. Pal, N; Bezdek, J.; and Tsao, E. Generalized clustering networks and Kohonen's self-organizing scheme. IEEE Trans. Neural Networks. 4:4(1993): 549-557.
 15. Zhang, Y. and Liu, Z. Self-splitting competitive learning: A new on-line clustering paradigm. IEEE Trans. Neural Networks. 13:2(2002): 369-380.
 16. Bradley, P.; Fayyad, U.; and Reina, C. Clustering very large databases using EM mixture models. Proc. 15th Int. Conf. Pattern Recognition. 2(2000): 27-80.
 17. Jain, A.K.; Duin, R.; and Mao, J. Stistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell. 22:1(2000): 4-37.
 18. Bezdek, J.C. Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum Press, 1981.
 19. Ester, M.; Kriegel, H.; Sander, J; and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD'96). (1996): 226-231.
 20. Comaniciu, D. and Meer, P. Distribution free decomposition of multivariate data. Pattern Analysis and Application. 2(1999): 22-30.

21. Yang, M. and Wu, K. A similarity-based robust clustering method. IEEE Trans. Pattern Anal. Mach. Intell. 26:4(2004): 434-448.
22. Ng, R. and Han, J. Efficient and effective clustering method for spatial data mining. Proc. 20th VLDB Conference. pp. 144-155. 1994.
23. MacQueen, J. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. pp. 281-297. 1967.
24. Lin, C. and Chen, M. Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. IEEE Trans. Knowledge and Data Engineering. 17:2(2005): 145-159.
25. Wu, S. and Chow, T.W.S. Self-organizing-map based clustering using a local clustering validity index. Neural Processing Letters. 17(2003): 253-271.
26. Kohonen, T. Self-Organizing Maps. New York: Springer-Verlag, 2001.
27. Kohonen, T.; Oja, E.; Simula, O.; Visa, A.; and Kangas, J. Engineering applications of the self-organizing map. Proc. of IEEE. 84:10(1996): 1358-1384.
28. Alahakoon, D.; Halgamuge, S.K.; and Srinivasan, B. Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Trans. Neural Networks. 11:3(2000): 601-614.
29. Pakkanen, J; Iivarinen, J; and Oja, E. The evolving tree—a novel self-organizing network for data analysis. Neural Processing Letters. 20(2004): 199-211.
30. Chow, T.W.S. and Wu, S. Cell-splitting grid: A self-creating and self-organizing neural network. Neurocomputing. 57(2004): 373-387.
31. Pakkanen, J; Iivarinen, J; and Oja, E. The Evolving Tree, a hierarchical tool for unsupervised data analysis. Proc. IJCNN 2005. pp. 1395-1399. 2005.

32. Pakkanen, J; Iivarinen, J; and Oja, E. The Evolving Tree—analysis and applications. IEEE Trans. Neural Networks. 17:3(2006): 591-603.
33. Xu, P.; Chang, C; and Paplinski, A. Self-organizing topological tree for online vector quantization and data clustering. IEEE Trans. Syst., Man, and Cybern. – Part B. 35:30(2005): 515-526.
34. Weisstein, E.W. Variation Coefficient. [Online]. Available from: <http://mathworld.wolfram.com/VariationCoefficient.html>



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Biography

Name: Ms. Ureerat WATTANACHON.

Date of Birth: 12th June, 1975.

Educations:

- Ph.D., Program in Computer Science, Department of Mathematics, Chulalongkorn University, Thailand, (October 2001 - October 2006).
- Ph.D. Visiting student, School of Information Technology Electrical and Engineering, University of Queensland, Brisbane, AUSTRALIA (February 2005 - July 2005).
- M.Sc. Program in Computer Science, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (June 1997 - March 2001).
- B.Sc. Program in Mathematics, Faculty of Science, Kasetsart University, Bangkok, Thailand (June 1993 - March 1997).

Publication papers:

- U. Wattanachon and C. Lursinsap. Class-Driven Self-Grouping Learning for Pattern Classification. *Proceedings of the 4th International Conference on Intelligent Technologies*, pp. 108-116, 2002.
- U. Wattanachon and C.Lursinsap. Agglomerative Hierarchical Clustering for Non-linear Data Analysis. *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2004)*, pp. 1420 - 1425, 2004.

Scholarship: The Development and Promotion for Science and Technology Talents Project (DPST) of Thailand.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย