Mining Arbitrary Shaped Clusters in Large Dataset

BIDYUT KUMAR PATRA

Mining Arbitrary Shaped Clusters in Large Dataset

A Thesis

Submitted in partial fulfillment of the

requirements for the Degree of

Doctor of Philosophy

by

Bidyut Kumar Patra

Under the Supervision of

Sukumar Nandi

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati – 781039

APRIL 2012

This thesis is dedicated to my parents and uncle.

Declaration

I hereby declare that,

1. The work contained in this thesis is original and has been done by myself under the supervision of my thesis advisor.
2. To the best of my knowledge this work has not been submitted elsewhere in any University for the award of any degree or diploma.
3. I have given appropriate credit to the original authors whenever I used materials (data, text or figures) from other sources.
4. Whenever I have quoted written material from other sources, I have put them under quotation marks and given due credit to the sources by citing the sources and giving required details in the references.
5. I have followed the norms and guidelines given in Ethical Code of Conduct of the Institute.

Date:

Bidyut Kumar Patra
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati- 781 039, India.

Certificate

This is to certify that this thesis entitled "**Mining Arbitrary Shaped Clusters in Large Dataset**", submitted by **Bidyut Kumar Patra**, a research student in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Date:

Place: IIT Guwahati, India.

Dr. Sukumar Nandi,
Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati- 781 039, India.

Acknowledgment

First and foremost, I would like to thank my thesis supervisor Prof. Sukumar Nandi for introducing me to the world of research. I am grateful to him for his careful guidance, encouragement during my thesis work. I am deeply indebted to him for making me analysis research articles. I have learnt a lot of things from him staring from academic to non academic matters.

I would like to express my gratitude to Prof. Vijay Iyenger, Visiting Faculty, IIT Guwahati for his critical comments on some of my work. I would like to thank Dr. Vijaya Saradhi, Faculty, IIT Guwahati for giving me time to discuss some technical issues related to my thesis work. I am thankful to Dr. P. Viswanath, ex-faculty, CSE, IIT Guwahti for his help and encouragement at the initial phase of my thesis work.

I would like to thank my Doctoral Committee members Dr. S. V. Rao, Dr. P. K. Das and Prof. Ratnajit Bhattacharjee for their valuable comments and suggestions to improve the quality of the work. I am extremely thankful to Dr. P. Bhaduri for allowing me to use research facilities available in the department.

My sincere thanks go to all office staffs Malakarda, Rakta, Prabin, Nobo for helping me in many occasions. I am also thankful to Nanu, Bhrigu for providing me a high end server for almost six months.

I would to like to express my heartiest thanks to Rajendra, with whom I shared office room in the department. We used to discuss many issues some technical, some non technical. I am indebted to Rajendra for his kind help and suggestions make me more stronger and confident in all aspects. I used to discuss with my fellow research scholar Suresh Suddhasil in many occasions. I am also thankful to them. I am thankful to

Neminath with whom I spent time discussing technical and political matters.

I would like to thank Bibhas Adhikari, who inspried me to learn Mathematics. I am thankful to my friends Bitta, Sushen-Soma, Kaushik, Avijit, Sounak (Nepal), Debasish, Barun who made my stay at IIT Guwahati pleasant.

I would like to express my sincere thanks to Arup sir who helped me in many ways (both in academically and non academically).

I am thankful to my fellow research scholars Mohanty, Mamata, Lipika, Mousumi. We used to discuss many topics during evening tea break. This discussion helped me to get refreshed time to time. I thank Suddasil for providing MATLAB code atleast in two occasions.

I would like to thank D. Satyanarayana who always encourages me to think big. I am also thankful to Tejmani Sir, Nityananda Sir for their valuable comments about research work.

I would like to thank Prof. D. K. Bhattacharjee, former HOD, CSE, Tezpur University for allowing me to work at IIT Guwahati during semester break and holidays. I am also thankful to present HOD, Dr. Utpal Sharma for providing me research facility at Tezpur University.

I am thankful to my friend S.R. Singh, faculty member in this department for his kind help in understanding the some technical matters. I would like to thank my friend Biswapati, Jaydeb for their constant encouragements.

I would like to express my gratitude to my father and uncle for their endless encouragements. I am also thankful to my brothers and sisters, Barda (Asish), Mejda (Pijush), Sejda (Prabitra), Bardi (Sikha), Mejdi (Dikha), Chotdi (Santon), Astu, Raju, Sur, Bulti and Lalan for their love and encouragement.

Finally, I would like to mention someone for whom I am able to finish my Ph.D. work. She is my wife. She always encourages me to do some big things. She always encourages me to think big. Love and affection to my little son encourage me to complete the writting early.

Contents

List of Tables

List of Figures

Abstract

Cluster analysis or clustering, groups objects into a meaningful manner, is an important task in pattern recognition, data mining, bio-informatics, image processing, information retrieval, intrusion detection system, *etc.* To meet demands of clustering activities in these domains, many clustering methods have been developed over the years. These clustering methods can be categorized from different perspectives. An important perspective is *algorithmic perspective*. From algorithmic perspective (view), clustering methods are classified into *distance based* and *density based* methods (algorithms). Naturally, clusters are *arbitrary shaped* and different *sizes* in a dataset. However, finding clusters of arbitrary shapes and sizes is a difficult (challenging) task, especially in large size dataset. This thesis focuses at finding clusters of arbitrary shapes and sizes using both distance based and density based algorithms in large dataset. Distance based clustering methods usually find convexed shaped clusters. The classical single-link is a distance based method, which can find clusters of arbitrary shapes and sizes. However, it scans dataset many times and has quadratic time and space complexity. This prohibits single-link to apply to large dataset. To address these issues, a distance based hybrid clustering method has been proposed, which is termed as *l*-SL method. It produces a flat clustering for arbitrary shaped clusters in large dataset. It has two stages. In the first stage, leaders clustering method is applied to a dataset to obtain many convex shaped small clusters. In the second stage, these convexed shaped clusters are merged using single-link method (given minimum distance criteria). Proposed *l*-SL method scans a dataset once and clustering results of *l*-SL method is close to the final clustering of that of the single-link method. An improvement of the *l*-SL method has also been proposed in order to produce same clustering results as that of the single-link clustering method. Proposed *l*-SL and its improved methods output a flat clustering of a given dataset. A new data summarization scheme termed as *data sphere* has been proposed, which exploits leaders method to acquire summary of a dataset. Proposed *data sphere* is utilized to work with hierarchical single-link method

in large dataset for generating a hierarchy of clustering. However, clustering results of *data sphere* based single-link method deteriorates at high compression ratio. To address this problem, tolerance rough set theory based data summarization scheme termed *rough bubble* has been proposed. Tolerance rough set theory is exploited to resolve uncertainty associated with cluster membership of leaders method, which collects directional distance statistics of each *rough bubble*. Main disadvantage of proposed distance based *l*-SL, improved *l*-SL, *data sphere* and *rough bubble* methods is that they cannot find arbitrary shaped clusters in presence of outliers, varying density clusters. To address these issues, a dynamic parameter termed *Nearest Neighbor Factor*(NNF) is introduced to determine relative position of an object in its neighborhood (local) region. A density based clustering method which uses NNF is introduced for finding clusters of arbitrary shapes, different sizes and variable density in a dataset.

Chapter 1

Introduction

We are living in digital world. Everyday individual person, business houses, various organizations are accumulating enormous quantities of digitized information/data. These massive size data need to be analyzed efficiently for better understanding behavior of data or different decision making process. Various techniques are devised for analyzing them automatically. These techniques are broadly classified into two categories: (i) confirmatory analysis or classification and (ii) exploratory or cluster analysis.

Confirmatory analysis classifies an object into one of the several predefined categories. The confirmatory or classification technique works as follows. The technique needs a set of objects with their categories, called *training set*. Let (x, y) be an element of a training set, where x is an object and y is the category of x. Very often, object x is called "seen" as y is known. Based on the information available of seen objects in the training set, technique finds a suitable model, which captures relationship between objects and their categories or classes. Finally, model assigns a class label to an unseen object (not belong to training set). The task of assigning a class label to an unseen object is supervised by a training set. Therefore, classification task is popularly known as *supervised learning*. Suppose, one wants to classify an animal into one of the three categories: Amphibians, Reptiles and Mammals. To perform this task, characteristics of animals with their categories are provided (*training set*). Based on the knowledge (*model*) extracted from the characteristics, one can predict whether an unknown animal belongs

to one of the three categories. In general, category can have finite discrete values. In the previous example, Amphibians, Reptiles and Mammals are the values of category. If classification task predicts a continuous value (y) for an unseen object, then it is called *regression*. Main drawback of the classification analysis is the non availability of suitable training data. It is not always easy to collect labeled data.

On the other hand, *cluster analysis* or *clustering* is to group objects into a meaningful manner. Objects in a group are alike compared to the objects in different groups. The basic difference between this approach of data analysis from earlier approach is that it is not guided by any training set, i.e., grouping of objects is performed without any supervision- *unsupervised learning*. This makes clustering task more challenging in nature. For examples, a set of animals are provided without mentioning the categories of individual animal. Here, task is to group animals based on alikeness (similarity) of their characteristics. Clustering task very often encounters the following challenges:

- How many groups need to be created in a given data. What can be the size of each group? Can an object belong to more than one group simultaneously?
- How the alikeness is measured between objects. Is it same to all applications?
- How can one measure "goodness" of a clustering task?

Nevertheless, clustering has been widely used in many application domains such as marketing [Shaw et al. (2001)] , image segmentation [Porter and Canagarajah (1996), Bowyer and Ahuja (1996)], geological mapping [Martelet et al. (2006)], bio-informatics [Datta and Datta (2003), Thalamuthu et al. (2006)], web mining [Joshi and Krishnapuram (1998), Lingras et al. (2004)], intrusion detection [Hubballi (2011)], *etc.*

Rest of the chapter is organized as follows. Section 1.1 mentions some important application domains where clustering technique is used frequently. Section 1.2 describes briefly major approaches to clustering task and motivations of the thesis. Contributions of the thesis are highlighted in section 1.4. Finally, organization of this thesis is discussed in section 1.5.

1.1 Applications of Clustering Technique

Goal of cluster analysis is to form groups of similar objects. Here, objects could be students at University, customers in a super market, biological genes, web pages at Internets, news articles, mobile consumers in a locality, investors at stock-exchanges, books in a library, satellite images, hand written characters, stars and planets in sky or patient records. Cluster analysis have been utilized in numerous application domains. Some of them are listed below.

Biology

Biologists in 18^{th} and 19^{th} centuries spent their lives in categorizing or classifying unseen plants and animals. Taxonomy of living creatures is an important aspect in biology. Clustering technique is used to find the taxonomy automatically. Evolutionary biologist Masatoshi Nei used clustering approach to create phylogenetic tree [Nei and Kumar (2000)]. Recently, clustering has been applied to genomic data to group functionally similar genes [Bolshakova et al. (2004), Azuaje (2002), Azuaje (2003), Aguilar-Ruiz and Azuaje (2004), Chen et al. (2004), Vinh and Epps (2009)].

Information Retrieval (IR)

Typically, search engines such as Google, Yahoo provide a list of search results in response to search string. User examines each title and snippet of search results carefully until s/he gets most relevant documents. This is a very tedious job, especially if a search word has different senses. However, search results can be clustered so that related documents appear together in a cluster and user can be asked to find relevant cluster(s) [Zeng et al. (2004)]. Thus, user can quickly browse relevant and desired information. Vivisimo search engine (`http://vivisimo.com/`) has this features.

Scatter/Gather technique in IR system utilize document clustering methods to group relevant documents together and separates them from non-relevant ones. In Scatter/Gather, clustering is applied repeatedly to the search results to refine the results [Hearst et al.

(1995), Hearst and Pedersen (1996)]. Other clustering based information retrieval approaches are found in [Altingovde et al. (2007), Altingovde et al. (2008)].

Market Research

Cluster analysis has been widely used in market segmentation. The goal of market segmentation is to group entities (people, markets, organizations) based on their common characteristics. Taxonomy of customers' behavior can be a tool for a better understanding of customers' choice and characteristics. Several researchers suggested to utilize cluster analysis for creating this taxonomy [Furse et al. (1982), Punj and Stewart (1983)]. Most market segmentation approaches find non-overlapping partitions of customers. However, it is found that a consumer can belong to more than one segments. The overlapping k-centroids clustering is developed for grouping customers [Chaturvedi et al. (1997)]. Fuzzy clustering methods are also used for market segmentation [Hruschka (1986)].

Image Segmentation

Image segmentation is an important part of image analysis and image understanding. Image segmentation partitions an input image into a number of homogeneous regions so that pixels in a region are more similar than the pixels in different regions. Clustering technique was first used in image segmentation in 1969 [Wacker (1969)]. The simplest approach is *thresholding*, which groups image pixels based on a scaler value of image intensity [Rosenfeld et al. (1969), Dunn et al. (1983)]. The thresholding concept is extended to several dimensions considering a pixel as a N dimensional vector. Clustering methods were applied to segment multispectral remote sensor imagery [Fu and Mui (1981)]. Jolion used clustering method for threshold selection in range and intensity image segmentation [Jolion et al. (1991)]. Real image consists of textured and smooth regions. Therefore, feature sets are different in each regions. Having extracted the features, a clustering method is applied to segment the both regions separately [Porter and Canagarajah (1996)].

Object and Character Recognition

Human beings can easily recognize objects in images, even-though objects are obtained from any arbitrary view point, objects can be in different sizes, scale in images. However, this task of object recognition in computer vision is very challenging. In object recognition system, multiples views of an objects are stored in training set and system is asked to recognize an unseen object. To speed up this process, clustering process play an important role [Lowe (2001)]. Hierarchical clustering (complete-link) is used to cluster the views of each object in the training set in order to make recognition process faster [Dorai and Jain (1995)]. Clustering is used in handwriting character recognition system. Writer-independent system needs a large amount of training set as writing varies from person to person. As the variability of the writing styles increases, it becomes more and more challenging task to distinguish one class from another. Clustering analysis can be applied to separate these disparate writing styles for each class. A variant of k-means clustering method called CLUSTER is used for this purpose [Jain and Dubes (1988)]

Climate

Defining or classifying climate zones based on different meteorological variables (monthly average, maximum and minimum temperatures, and monthly total precipitation) is an important issue in climatological research. Unal used clustering methods to find homogeneous climate zones of Turkey considering both the variables temperature and precipitation together [Unal et al. (2003)]. Precipitation climate of Iran was regionalized using clustering technique [Dinpashoh et al. (2004)]. Mimmack analyzed the effects of Euclidean and Mahalanobis distance measures to define (group) climate regions using hierarchical clustering methods [Mimmack et al. (2001)]. This approach uses only one meteorological parameter (monthly rainfall) for this task. Tropical rainfall stations are also classified using clustering approach [Jackson and Weinand (1995)].

Psychology

Psychologists very often group a set of people into a number of subgroups based on their profiles. A psychological test can be conducted on all people in a group. Therefore, clustering approaches can be a useful tool for this purpose. Another approach is commonly used in psychological community is to identify a characteristics (feature) over a group of people. Clustering methods can be used to find group of people, who change their behavior over different occasions [Borgen and Barnet (1987)].

Miscellaneous

Clustering methods are also used in other applications like social network analysis [Breiger et al. (1975)], health care data analysis, astronomical image and data analysis [Starck and Murtagh (2002)], intrusion detection system [Burbeck and Nadjm-Tehrani (2004)], etc.

1.2 Major Approaches to Clustering Technique

Various application domains have been motivating to develop different clustering methods since mid 50s of the last century. Therefore, cluster analysis literature is very rich and vast [Jain et al. (1999), Jain et al. (2000), Xu and Wunsch (2005), Jain (2010)]. These methods can be classified from different viewpoints. One important view on which clustering methods can be classified is *algorithmic view*. Algorithmic view classifies clustering methods based on the types of algorithms used - *distance based method* and *density based method*.

1.2.1 Distance based clustering method

Distance based clustering methods use inter-object distance information for clustering task. In this approach, methods optimize a global objective (criteria), which is based on the inter-object distances. Popular examples of distance based clustering methods are k-means [Steinhaus (1956), MacQueen (1967), Lloyd (1982)], CLARA , CLARANS, leaders

[Hartigan (1975), Spath (1980), Mitra and Acharya (2003)], singe-link [Sneath and Sokal (1973)], complete-link [King (1967)] and many more.

Distance based methods are again divided based on the type of clustering results produced by them as follows.

- **Partitional clustering method :** Partitional clustering methods create a single clustering (flat clustering) of a dataset. Partitional clustering method divides a dataset into a number of pre-specified clusters. Let C_1 and C_2 be two clusters in a clustering of a dataset. Then, partitional clustering methods follow these constraints:

$$(i)\ C_1 \nsubseteq C_2 \text{ or } C_1 \nsupseteq C_2, \quad (ii)\ C_1 \cap C_2 = \emptyset$$

  Most popular distance based partitional clustering method is k-means [Steinhaus (1956), Lloyd (1982)]. The k-means takes k points as the input. The k-means minimizes *Sum of Squared Euclidean distances* between patterns and theirs centroids. The k-means runs in linear time with the size of the dataset. CLARA, CLARANS are different variant of k-means clustering methods.

  Leaders [Hartigan (1975)] is another distance based method, which scans a dataset once only. It has linear time and space complexity. Therefore, it is suitable for large dataset.

  Fuzzy c-means [Dun (1973)], rough k-means[Lingras and West (2004)] are fuzzy and rough set based clustering approaches, respectively. These methods can be used to find overlapped clusters in various domains like information retrieval, document clustering. Fuzzy and rough clustering approaches violate a constraint stated above: $C_1 \cap C_2 \neq \emptyset$, where C_1 and C_2 are two clusters of the partitional clustering.

- **Hierarchical clustering method :** Partitional clustering methods cannot produce hierarchical structures present in data. Hierarchical clustering methods creates a numerical taxonomy of a given dataset, which represents hierarchical structures of

the dataset. Hierarchical clustering method can be seen as a type of clustering method which creates a sequence of partitional clusterings of a dataset. Based on the requirements of application domains, a clustering is considered as the final clustering of a given dataset. Hierarchical clustering methods are known to produce better clustering results than partitional clustering method [Lin and Chen (2005)]. Let C_1 be a cluster in a clustering of a dataset and C_2 be a cluster in another clustering of the same dataset. Then, hierarchical clustering satisfies one of the following two conditions:

$$(i)\ C_1 \subseteq C_2 \text{ or } C_1 \supseteq C_2, \quad (ii) C_1 \cap C_2 = \emptyset$$

Hierarchical clusterings of a dataset can be obtained either in *top-down* or *bottom-up* approaches. Initially, all objects are considered in a cluster in top-down approach. Then, recursively, each cluster is divided into smaller size clusters until each one contains single object. Bottom-up approach considers each object in a dataset as a singleton cluster and merges similar clusters recursively until all objects are in a single cluster. Bottom-up approach is widely used to produce hierarchical taxonomy of a dataset.

Popular examples of distance based (bottom-up) hierarchical methods are single-link, complete-link, average-link clustering methods. These three clustering methods differ in measuring similarity/dissimilarity between a pair of clusters. Each of these methods has quadratic time and space complexity with the size of the dataset.

1.2.2 Density based clustering method

Most of the distance based clustering methods cannot find clusters of arbitrary shapes and size in a dataset. However, density based clustering methods can find arbitrary shaped clusters. Density based clustering methods form clusters based on the density distribution of the clusters. Basic assumption of this approach is that clusters are high density regions

in a given feature space [1] separated by low density regions. Therefore, density based approach defines an appropriate function (mechanism) to identify high density or low density region in the feature space. It uses local information of objects in the feature space unlike distance based methods. Several methods have been developed over the years. DBSCAN [Ester et al. (1996)] is a very widely used density based clustering method. DenClue [Hinneburg and Keim (1998)], GRIDCLUS [Schikuta (1996)], OPTICS [Ankerst et al. (1999)], NBC [Zhou et al. (2005)] are few examples of density based approach.

In the same line of distance based clustering approach, these methods are again classified as follow.

- **Partitional method :** Partitional clustering methods divide a given feature space into dense and non-dense regions based on the density of objects in the space. Dense regions are treated as clusters. DBSCAN, DenClue, NBC are density based partitional clustering methods.

  DBSCAN considers a hyper-sphere of pre-specified radius around each object and counts the number of objects in the hyper-sphere. If the count is more than a pre-specified number, then the object is considered as *dense object*. Adjacent dense objects in the feature space form a cluster. Two clusters are separated by less dense regions.

  DenClue uses a function for measuring density of an object. A sequence of objects with local (maxima) density more than a pre-specified threshold is considered as a cluster.

  NBC uses K nearest neighbors and reverse K nearest neighbors information of an object to identify dense objects. It merges adjacent dense objects to form a cluster.

- **Hierarchical method :** Partitional clustering methods cannot find nested clusters present in data whereas hierarchical methods can find them. OPTICS, GRIDCLUS

[1] An object is characterized by a number of attributes called features. An object can be thought of a point in a N dimensional Euclidean space, where a dimension corresponds to a feature.

are classical examples of density based hierarchical clustering method. OPTICS is the hierarchical version of DBSCAN method.

1.3 Major issues, approaches to clustering large dataset, and motivation of the thesis

Naturally, clusters are *arbitrary shaped* and different *sizes* in a data. However, finding clusters of arbitrary shape and sizes remains a challenging task, especially in large dataset. Nowadays, many applications like bio-informatics, information retrieval, image analysis, social network analysis gather large size data. Very often these voluminous data need to be clustered for better understanding or further analysis the domains. However, classical clustering approaches fail to handle them efficiently. The works in this thesis address the problems of finding "natural" clusters (arbitrary shaped and size) in large size data. Thesis uses both approaches (distance based and density based) to find clusters in large dataset.

In this section, major issues and approaches to clustering large dataset are discussed along with motivation of the thesis.

1.3.1 Issues with distance based clustering approach

Distance based clustering approach uses only inter-objects distances. It optimizes a global objective function explicitly or implicitly. Distance based partitional clustering methods like k-means, leaders, CLARA find convexed shaped clusters. The k-means, CLARA scan the dataset many times before they converge. Therefore, these methods are not suitable for clustering large dataset. Leaders can be suitable for large dataset as it scans the dataset only once. However, it cannot find arbitrary shaped clusters.

Single-link is a distance based hierarchical method, which can find arbitrary shaped clusters. However, single-link cannot be used for clustering large dataset due to the following drawbacks:

- Single-link has time and space complexity of $O(n^2)$, where n is the size of the dataset.

- It scans the dataset many times for computing all pair-object distances.

These two drawbacks can be addressed to some extent by first applying a linear clustering (k-means) method to a dataset and subsequently single-link is applied to the output of linear clustering [Murty and Krishna (1981), Wong (1982)]. However, these methods cannot be suitable for large dataset as k-means scans a dataset many time before convergence. Therefore, a clustering method is to be developed which scans a datset once and further scans can be allowed to refine the clustering results.

1.3.2 Issues with density based clustering approach

Density based clustering methods can find clusters of arbitrary shapes and sizes. It uses local information of an object to find density of the object in the feature space. DBSCAN [Ester et al. (1996)] is a very popular density based method. It has following drawbacks:

- DBSCAN needs two important parameters, i.e., radius of hyper-sphere around an object and minimum number of objects in the hyper-sphere for which object is considered as dense object. Finding the both parameters in real world dataset is a difficult task.

- It cannot find clusters if variability of density of clusters is very high in a dataset.

- The method has time complexity of $O(n^2N)$, where n and N are the size and dimension of the dataset, respectively.

1.3.3 Major approaches to clustering large dataset

Many clustering approaches have been proposed to tackle large size data in domains like gene clustering, content based image retrieval, document clustering, earth observation science. These approaches are mainly classified into the following listed groups [Jain (2010)]. It may be noted that we restrict our discussion to the clustering approaches which use single processor during computations.

- **Sampling Based Approach:** In this approach, a subset of objects (samples) are selected randomly, which is considered to be the representatives of the whole dataset. Subsequently, an expensive clustering method is applied to the subset. Finally, clusters of random samples are transferred to clusters of the dataset. CLARANS, CURE [Guha et al. (2001)] are examples of this approach. The clustering results are dependent on the sample size. With less sample size clustering results deteriorate significantly.

- **Hybrid Clustering:** Hierarchical clustering methods are found to produce better clustering results than partitional clustering methods. However, hierarchical methods suffer from high computational and storage burden whereas most of the partitional clustering methods runs in linear time with the size of the dataset. Therefore, hybridization of partitional and hierarchical clustering methods are used to work with large dataset [Murty and Krishna (1981), Lin and Chen (2005), Liu et al. (2009)]. In this approach, initially a less expensive (linear) partitional clustering method is applied to the whole dataset. Subsequently, a computationally expensive hierarchical clustering method is applied to the outputs of the partitional method. This leads to a large reduction in run time of the hierarchical clustering methods while compromising a very little clustering quality.

  Thesis has a contribution in this direction. A hybrid clustering method is proposed for mining arbitrary shaped clusters in large dataset.

- **Data Summarization:** This approach creates a summary of a large size dataset and this summary is intelligently used to scale up expensive clustering methods. Generally, size of the summary set is significantly less compared to the size of the original dataset. CF-tree in BIRCH [Zhang et al. (1996)] clustering approach is used as the summary of the whole dataset. Data Bubbles [Breunig et al. (2000, 2001)] is another data summarization technique proposed to speed up hierarchical clustering method.

New data summarization schemes are developed in this thesis to speed up hierarchical single-link method in large size data.

- **Nearest Neighbor (NN) Search:** Density clustering methods like DBSCAN needs to find out neighbors objects (range query) of each data object in the dataset. Therefore, DBSCAN uses tree-based indexing technique. However, tree-based indexing technique does not perform well with the increase of dimensions. LSH-link uses *locality sensitive hashing* to search nearest neighbor, which speeds single-link method up in large dataset. NBC [Zhou et al. (2005)] method uses VA file [Weber et al. (1998)] approach to speed up the K nearest neighbor search. Recently, M. Kryszkiewicz et al. [Kryszkiewicz and Lasek (2010a)] showed that VA file does not perform well in high dimensions data. They introduced TI-K-Neighborhood-Index scheme for finding K nearest neighbors very quickly [Kryszkiewicz and Lasek (2010a)].

  Thesis has one contribution in this direction. A density based clustering method, which uses TI-K-Neighborhood-Index is proposed in this thesis.

1.4 Contributions of the Thesis

In the thesis, we propose three distance based and one density based clustering methods for mining arbitrary shaped clusters in large dataset. Contributions of the thesis are summarized as follow.

- A hybrid clustering method termed as *l*-SL method is proposed. Proposed hybrid method has two stages. In the first stage leaders clustering method is applied to a given dataset to obtain many convex shaped small clusters. In the second stage, these convexed shaped clusters are merged using the single-link method (minimum distance stopping criteria). Proposed method scans the dataset once only. Proposed *l*-SL is significantly faster than that of the single-link method and clustering results produced by *l*-SL method is comparable with the final clustering of that of the single-link method. Method is suitable in large dataset. However, clustering results

of proposed l-SL is not exactly same as that of the single-link method. To compensate the clustering results, an improvement of the l-SL method is also proposed. Clustering results produced by this improved method are analyzed (theoretically and experimentally). Improved method is also significantly faster than the classical single-link method.

- A new data summarization scheme termed *data sphere* is proposed to speed up hierarchical single-link method. Leaders clustering is utilized to obtain a summary of a given dataset. A leader with summarized information of its followers is referred to as *data sphere*. A similarity measure between a pair of *data sphere*s is introduced to apply agglomerative single-link method to the summarized set. This produces a hierarchy of clustering, which is comparable with the hierarchy produced by classical single-link method. Proposed method scans the dataset twice only. To further speed up proposed summarization scheme, a technique is introduced to reduce a large number of distance computations in leaders method.

- A Tolerance Rough Set Model (TRSM) based summarization scheme termed *rough bubble* is proposed to address drawbacks of the summarization scheme proposed in [Zhou and Sander (2003)]. Proposed summarization scheme has two steps. In the first step, leaders method is modified (termed TI-LEADER) to quickly find representative patterns of a dataset. In the second stage, classical leaders method is used to collect sufficient statistics of the dataset. Proposed TRSM based summarization is combined with hierarchical single-link for clustering large dataset. Clustering results are compared with the classical single-link method and data bubble [Zhou and Sander (2003)] based single-link method.

- A density based clustering method, which can detect clusters of arbitrary shapes, sizes and different densities is proposed. A novel parameter termed *Nearest Neighbor Factor*(NNF) is introduced to determine relative 'importance' (position) of an object in its neighborhood region (K neighborhood). Parameter NNF accounts closeness and K neighborhood density of a pattern in the feature space. Based on

the relative position of a point, the proposed method expands a cluster recursively or declares the point as an outlier. Experiments are conducted to show the effectiveness of the proposed method. It outperforms classical density based clustering method DBSCAN and recently proposed TI-K-Neighborhood-Index supported NBC method.

1.5 Organization of the Thesis

The dissertation is organized as follows.

Chapter 2: This chapter presents a survey of clustering methods and background of the present work.

Contributions of the thesis are discussed in subsequent chapters, Chapter 3 to Chapter 6. Conclusions and future research directions are discussed in Chapter 7.

Chapter 3: This chapter discusses two hybrid clustering methods *l*-SL and *al*-SL method for mining arbitrary shaped clusters.

Chapter 4: In this chapter a new data summarization scheme *data sphere* is proposed to speed up classical single-link method.

Chapter 5: This chapter deals with tolerance rough set based summarization scheme which can identify cluster separation present in a summarized unit.

Chapter 6: A density based clustering method, which can identify clusters of arbitrary shapes, size with variable densities is proposed in this chapter.

Chapter 7: This chapter summarizes overall contributions and discusses a future research directions of the thesis.

Chapter 2

Background and related work

2.1 Introduction

Cluster analysis is an important and challenging task in many fields of science, engineering and social science. Cluster analysis or clustering can be defined as follows. Let $\mathcal{D} = \{x_1, x_2, x_3, \ldots, x_n\}$ be a set of n patterns, called dataset, where each x_i is a point in N-dimensional space. Each x_i is a physical or an abstract object. Clustering activity is to find groups of patterns, called clusters in $\mathcal{D}$, in such a way that *similarities* between patterns in a cluster are high while similarities between patterns in different clusters are low. Similarity between a pair of patterns is a function of attributes or features, which describe the patterns. For example, an employee in an organization can be a pattern. It is described by a number of features such as employee id, name, age, sex, department, grade, phone number, etc. Features of a pattern can be two types-*Quantitative* and *Qualitative*. Quantitative features can have real or integers values. In the previous example, age is a quantitative feature and all others are qualitative features. Quantitative features can be subdivided into three types - (a) continuous, (b) discrete and (c) interval types. Qualitative features have a finite set of values. If values in the finite set can be ordered in a meaningful sequence, then it is called *ordinal*, otherwise it is called *nominal* features. Grade is an ordinal feature as grades of employees can be ordered, i.e., teaching staffs in the University can be ordered as Assistant Professor, Associate Professor and Professor.

In the employee example, sex, department, phone number are nominal features.

2.2 Similarity Measures

Clustering methods find clusters of objects based on similarities between objects in a dataset. Therefore, methods require a measure of *similarity* between objects in the dataset. Similarity between a pair of patterns x and y in $\mathcal{D}$ is a mapping, $sim(x, y) : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1]$. Closer the value of $sim(.)$ is to 1, higher the similarity and it is 1, when $x = y$. Based on domains and features types, various similarity measures have been evolved over the years [Cha (2007)]. Commonly used similarity measures are discussed next.

Simple Matching Coefficient (SMC): SMC is used to calculate similarity between two binary vectors. Let x and y be two N-dimensional binary vectors, i.e. $x, y \in \{0, 1\}^N$. SMC between x and y is :

$$SMC = \frac{\sum_{i=1}^{N} t \mid t = \begin{cases} 1 & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases}}{N}, \tag{2.1}$$

where, x_i and y_i are the i^{th} feature values of x and y, respectively. SMC weights both attributes (0/1) equally.

Jaccard Coefficient: Jaccard Coefficient gives important to positive value (1) only. For examples, positive outcome of a disease test is more important than negative (0) outcome. Therefore, it is called asymmetric measure. Jaccard Coefficient between x and y is calculated using equation (2.2).

$$J = \frac{a_{11}}{N - a_{00}}, \tag{2.2}$$

where $a_{11} = \sum_{i=1}^{N} t \mid t = \begin{cases} 1 & \text{if } x_i = y_i = 1 \\ 0 & \text{otherwise} \end{cases}$, $a_{00} = \sum_{i=1}^{N} t \mid t = \begin{cases} 1 & \text{if } x_i = y_i = 0 \\ 0 & \text{otherwise} \end{cases}$

Let $x = (1, 0, 0, 1, 1)$ and $y = (0, 1, 0, 1, 0)$ be two binary vectors, respectively. Then, SMC and Jaccard Coefficient are as follow.

$SMC = \frac{2}{5}$; Jaccard Coefficient $= \frac{1}{4}$

Cosine Similarity: In some applications like information retrieval, text mining, a document is represented as a vector, where value of a feature is the frequency of a particular word appeared in the document. Having preprocessed documents, important words form the features set. Generally, a document vector contains many zero-valued features and a few non zero-valued. Therefore, symmetric measure is not used to compute similarity between documents like Jaccard Coefficient. On the other hand similarity measure must handle non binary features. *Cosine similarity* is widely used for this purpose. Let x and y be two document-vectors. Similarity between x and y are expressed as cosine of the angle between them (equation (2.3)).

$$cosine(x, y) = \frac{x \bullet y}{||x||\ ||y||} \tag{2.3}$$

where, $\bullet$ is the dot product and $||.||$ is L_2-norm of x, y. Consider $x = (3, 0, 2, 0, 0, 1)$ and $y = (2, 1, 3, 0, 1, 0)$ are two documents. The similarity between them is given below

$$cosine(x, y) = \frac{6 + 0 + 6 + 0 + 0 + 0}{\sqrt{3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 1^2}\ \sqrt{2^2 + 1^2 + 3^2 + 0^2 + 1^2 + 0^2}} = \frac{12}{\sqrt{13}\ \sqrt{13}}$$

Tanimoto Coefficient: A variant of cosine similarity called *Tanimoto Coefficient* is used in information retrieval, text mining, biological taxonomy. Similarity between x and y vectors can be computed using following equation:

$$Tan(x, y) = \frac{x \bullet y}{||x||^2 + ||y||^2 - x \bullet y} \tag{2.4}$$

Tanimoto Coefficient can also be used to compute similarity between a pair of sets. Let

A and B be two sets. Tanimoto Coefficient between A and B is the ratio of number of elements shared by both sets and total numbers of elements in sets A and B. Formally,

$$Tan(A, B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid} \tag{2.5}$$

Similarity between a pair of patterns x and y with nominal attributes depends on the number of attributes in which they match and it is calculated using equation (2.6) [Stanfill and Waltz (1986)].

$$\sum_i^N t = \begin{cases} 1 & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases}$$
$$S(x, y) = \frac{}{N} \tag{2.6}$$

Let $x = (CSE, \quad Professor)$ and $y = (ECE, \quad Professor)$ be two employees with two nominal features-department and grade. Similarity between x and y is $S(x, y) = \frac{0+1}{2}$.

One major drawback of this measure (equation (2.6)) is that it does not account the number of different states (values) an attribute can have in a dataset. Recently, data-driven similarity measures have been introduced for nominal attributes. In this measures, frequency distribution of different attributes states are taken into account in a given dataset to compute similarity between patterns. Some data-driven similarity measures are reported in [Boriah et al. (2008)].

Values of an ordinal attribute need to be mapped to subset of natural numbers in order to measure similarity between objects with ordinal attributes. Let f be an ordinal attribute, which can have M different values. These values can be ordered and mapped them to $\{1, \ldots, M\}$. Consider x_f and y_f are two values of f of two objects. Then, similarity between them is:

$$1 - \frac{|x_f^M - y_f^M|}{N}, \tag{2.7}$$

where, $x_f^M, y_f^M \in \{1, \ldots, M\}$, which are mapped values of x_f and y_f, respectively. Equation (2.7) can be generalized for objects with more than one ordinal attributes.

Many clustering methods use *dissimilarity* measures to find clusters instead of similarity measure. Dissimilarity between patterns in a cluster is less compared to the dissimilarity between patterns in different clusters (as opposed to similarity). Distance between a pair of patterns is often used as the dissimilarity measure in many clustering methods. Euclidean distance is widely used by machine learning, data mining communities for continuous data. Euclidean distance between a pair of N dimensional points (patterns) can be expressed as follow.

$$d(x,y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{2.8}$$

Generalization of Euclidean distance is known as Minkowski (equation (2.9)) distance.

$$L_p = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p} \tag{2.9}$$

This is referred to as L_p norm. If $p = 1$, we call it as L_1 norm or city block distance. Similarly, Euclidean distance is called L_2 norm.

Euclidean distance has some well known properties, which are given below. Let d be the Euclidean distance between a pair of objects in $\mathcal{D}$. Then d satisfies certain properties. For three patterns $x, y, z \in \mathcal{D}$,

- **Non-negativity:** $d(x,y) \geq 0$
- **Reflexivity:** $d(x,y) = 0$, if $x = y$
- **Symmetry:** $d(x,y) = d(y,x)$
- **Triangle inequality:** $d(x,y) + d(y,z) \geq d(x,z)$

A distance function that satisfy above all properties is called *metric* distance. However, Euclidean distance cannot be used when features have different scales and they are correlated among themselves. In this scenario, *Mahalanobis distance* [Mahalanobis (1936)] is very useful. It considers distribution of entire dataset. Mahalanobis distance between x and y

$$d_M(x,y) = (x-y)\sum^{-1}(x-y)^T,$$

where, $\sum$ is the covariance matrix of the dataset. Mahalanobis distance has time complexity of $O(nN^2)$, where n is the size of the dataset and N is the dimension of the dataset. Therefore, this method is not suitable in large dataset. Euclidean distance takes $O(N)$ times to compute distance between a pair of patterns. Euclidean distance is independent of underlying dataset. Clustering methods proposed in this thesis use Euclidean distance as the dissimilarity measure. Thesis exploits "Symmetry" and "Triangle inequality" properties of metric space to avoid certain distance computations while clustering large dataset.

Clustering methods may use non-metric distance function. It may be noted that any similarity measure can be converted to corresponding dissimilarity measure using equation (2.10). Let S be the similarity between a pair of patterns (obtained using any similarity measures). Then, dissimilarity dis can be

$$dis = 1 - S \tag{2.10}$$

A clustering method either uses similarity or dissimilarity or both depending on the application domain. Next section discusses different clustering methods, which are developed for various application domains.

2.3 Types of Clustering Methods

The essence of many fields such as statistics, pattern recognition, information retrieval, machine learning, data mining, psychology and other social sciences is to find meaningful groups of objects. Cluster analysis has been played an important role in these fields since mid 50s of the last century [Jain and Dubes (1988)]. Therefore, many clustering methods have been emerged for automatically finding meaningful groups or clusters. One can categorize these clustering methods from different perspectives, which are discussed next [Jain et al. (1999) Januzaj et al. (2003), Han and Kamber (2005), Tan et al. (2009)]. However, it is very difficult to give crisp categorization of clustering methods because

methods in one category may share some characteristic from other categories.

Partitional and Hierarchical : Clustering methods can be divided on the basis of clustering results *viz., partitional clustering* and *hierarchical clustering* methods. Partitional clustering methods produce a single clustering (flat clustering) while hierarchical clustering methods produce nested clusterings of a dataset. Most of the partitional clustering methods need the desired number of clusters as the prior knowledge. Hierarchical or taxonomic structures of data are discovered through hierarchical clustering methods.

Distance Based and Density Based: Clustering methods use optimization technique to produce clustering results. Some clustering methods uses global criteria, which is the function of distance between patterns in a dataset. These methods are called distance based method. On the other hand, density based clustering method optimize local criteria, which is based on density distribution of the dataset. In this approach, a densed region is considered as a cluster in the feature space. Both distance and density based methods can produce flat as well as nested clusterings of data.

Hard and Soft : Clustering methods, which assign a pattern to a cluster is called hard clustering. Here, cluster boundary is crisp (hard or rigid) and clusters are well separated. However, a pattern may be required to place in more than one clusters simultaneously in many applications. Here, clusters are overleaped, i.e. some patterns belong to more than one clusters. A different type of clustering method is proposed to handle these applications. This is called soft clustering approach.

Complete and Partial: Complete clustering methods assign each and every pattern to a cluster. Partial clustering could not assign some of the patterns to any clusters because these patterns do not belong to any meaningful groups in the data.

In the next subsections, we review widely used clustering methods with emphasizing on the methods which can detect arbitrary shaped clusters.

2.3.1 k-means clustering

The k-means [Steinhaus (1956), MacQueen (1967), Lloyd (1982)] is a distance based partitional clustering method. The k-means is very popular clustering method and it is

used in many application domains. It is an iterative method. The k-means assumes that the desired number of clusters, k is a user specified parameter. Initially, method selects k patterns randomly from dataset and treats them initial centroids of k clusters. Centroid of a cluster is the center point of a cluster. It needs not be a pattern in the given dataset. Each pattern of the dataset is then assigned to the closest centroids. Patterns assigned to a centroid form a new cluster and centroid of each of the k clusters are recalculated. The step of assignments to and calculation of centroid is repeated until all centroids remain unchanged in consecutive iterations. The k-means optimizes **Sum of Squared Error (SSE)** defined as follows. Let $C_1, \ldots, C_k$ be k clusters of the dataset. Then,

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{|C_j|} ||x_i - x^j||^2, \text{ where } x_i \in C_j, \quad x^j = \frac{1}{|C_j|} \sum_{i=1}^{|C_j|} x_i. \tag{2.11}$$

The k-means has time complexity of $O(I * k * n)$, where I is the number of iteration, n is the size of the dataset. Since, I is a small number with compared to the n and $k << n$, the method runs linearly in the size of the dataset. The space requirement of the method is $O(n)$.

However, it has many drawbacks:

i) It cannot detect noise point or outliers.

ii) It can find only convexed shaped clusters.

iii) It is applicable to only numeric dataset (vector space),

iv) With different initial points, it produces different clustering results.

There are many improvements of this basic k-means method [BALL and HALL (1967), Astrahan. (1970), Arthur and Vassilvitskii (2007)]. k-medoids is a variation of k-means method. Here, a medoid represents a cluster. A medoid is a cluster point, which locates close to center of the cluster. It can be applied to non-numeric dataset. However, it is computationally more expensive than the k-means method. Other variation of k-means method are CLARA [Kaufman and Rousseeuw (1990)], CLARANS [Ng and Han (2002)].

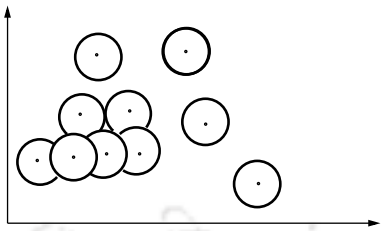


Figure 2.1: Leaders method generates spherical and semi-spherical clusters

2.3.2 Leaders clustering

There exists a category of clustering method called sequential algorithm. These methods are fast and create a single clustering of the dataset [Theodoridis and Koutroumbas (2006)]. Leaders clustering method [Hartigan (1975), Spath (1980), Jain et al. (1999), Mitra and Acharya (2003)] is one of this type. It is a distance based clustering method. It scans the dataset only once. Leaders clustering method has also been used in preclustering phase in data mining applications [Asharaf and Murty (2004)].

For a given threshold distance τ, it produces a single clustering of a dataset $\mathcal{D}$. Each cluster is represented by a pattern called *leader*. It produces a set of leaders $\mathcal{L}$ as follows. First pattern x_1 in the dataset is considered as the first leader in $\mathcal{L}$. For rest of the pattern $x \in \mathcal{D} \setminus \{x_1\}$, if there is a leader $l \in \mathcal{L}$ such that $||x - l|| <= \tau$, then x is assigned to the leader l. In this case, x is considered as a *follower* of the leader l. If no such leader is found in the leader set, then x is added as a new leader to $\mathcal{L}$. The leaders method is given in Algorithm 2.1. Main advantages of this method is that (i) it scans a dataset once, (ii) it can be applicable to any dataset where notion of distance is defined and (iii) it is an incremental algorithm. However, it has the following drawbacks.

- It can find only convex shaped clusters in data (Figure 2.1).
- Leaders method produces different clustering results for two different scanning orders of the same dataset,i.e., clustering results are ordered dependent.
- Similarity (distance) between patterns (followers) of different clusters (leaders) may be more (less) than corresponding leaders. Let x and y be two followers of two

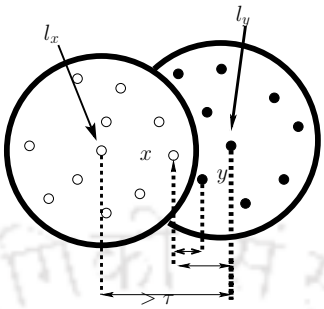


Figure 2.2: Distance between followers (x, y) of leaders l_x and l_y is less than τ

distinct leaders l_x and l_y, respectively. Distance between two leaders l_x and l_y is always more than τ, however distance between x and l_y or y and l_x or x and y may be less than τ. (Figure 2.2).

Complexity:

1. The time complexity of leaders clustering is $O(mn)$, where $m = \mid \mathcal{L} \mid$.

2. The space complexity is $O(m)$, if only leaders are stored; otherwise $O(n)$.

Algorithm 2.1 Leaders ($\mathcal{D}$, τ)

{$\mathcal{D}$ is a dataset and τ is the threshold distance of leaders clustering method}

$\mathcal{L} = \{x_1\}$; {x_1 is the first pattern in $\mathcal{D}$}
for each $x \in \mathcal{D} \setminus \{x_1\}$ **do**
  for each $l \in \mathcal{L}$ **do**
    if $||x - l|| \leq \tau$ **then**
      x is added as a follower of l; break
    end if
  end for
  if x is not follower of any leader **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
  end if
end for
Output $\mathcal{L}$.

Basic Sequential Algorithmic Scheme (BSAS), modified BSAS, Max-min algorithm, Two Threshold Sequential Algorithmic Scheme (TTSAS) [Juan and Vidal (2000)],

[Theodoridis and Koutroumbas (2006)] are variants of the leaders clustering method.

BSAS needs two user specified parameters: distance threshold (τ) and number of clusters (k). For each pattern x, method finds closest existing cluster C_{min}. If distance between x and C_{min} is more than τ and number of existing cluster is less than k, then x form a new cluster; otherwise, x is merged with C_{min} cluster. This scheme scans the dataset only once. One major drawback of BSAS scheme is that a pattern may not be assigned to overall closest cluster as scheme takes decision prior to the formation of final clusters.

Modified BSAS (MBSAS) is proposed to overcome the drawback stated above. MBSAS scans the dataset twice. In the first scan, complete set of cluster representatives $\mathcal{L}$, which is initialized with first pattern of the dataset are obtained as follows. For a point x, if distance from closest cluster-representative is more than τ, then x becomes a new cluster representative and added to $\mathcal{L}$. In the second scan, patterns are assigned to the closest cluster.

Max-min algorithm [Juan and Vidal (2000)] improves cluster selection process of MBSAS in the first phase. It works as follows. Let W be the set of selected points for clusters till the current iteration. For each $x \in \mathcal{D} \setminus W$, the scheme finds closest point $l_x \in W$ with distance d_x. Then it selects a point $x' \in \mathcal{D} \setminus W$ with $d_{x'}$ such that $d_{x'} = \max\{||x - l_x|| \mid x \in \mathcal{D} \setminus W, l_x \text{ is closest point in } W \text{ from } x\}$. If $d_{x'}$ is greater than τ, x' becomes new cluster and added to W. Otherwise, W is considered as final set of cluster points. Finally, each pattern in $\mathcal{D}$ is assigned to closest cluster in the second stage.

Leaders, BSAS, MBSAS, Max-min schemes are sensitive to the scanning order of the dataset. A Two-Threshold Sequential Algorithmic Scheme, which is less sensitive to the scanning order is proposed by introducing one more distance threshold (TTSAS) [Trahanias and Skordalakis (1989)]. The scheme scans a dataset many times. Let τ_1 and τ_2 ($> \tau_1$) be the two distance-threshold of TTSAS. In the first pass, each pattern x is checked whether distance between x and closest cluster C_x is less than τ_1 or not. If closest cluster-distance is less than τ_1, x is assigned to C_x. Otherwise, if closest cluster-distance is more than τ_2, x becomes a new cluster of $\mathcal{D}$. If closest cluster-distance is more than

τ_1 and less than τ_2, membership decision for x is deferred to later pass. In subsequent passes, first each unassigned pattern is checked for possible merging with closest cluster (if closest cluster-distance is less than τ_1) or forms a new cluster like first pass. If in any pass neither a single pattern is assigned nor a new cluster is formed, then an unassigned pattern creates a new cluster in the staring of next pass. If there is no unassigned pattern, the scheme terminates. The TTSAS may scan a dataset more than once before producing final output.

Leaders and its different variants are suitable for finding convexed shaped clusters in a dataset. However, these cannot identify nested clusters in a dataset. Hierarchical clustering methods can find nested clusters in a dataset. Popular hierarchical clustering methods are discussed next.

2.3.3 Hierarchical clustering method

Hierarchical clustering methods create a sequence of clusterings π_1, π_2, π_3, $\ldots$ π_i $\ldots$ π_p of $\mathcal{D}$. Main advantage of the hierarchical clustering method is that it can produce inherent nested structures (hierarchical) of clusters in a data. These hierarchical structures can be represented by a special tree called *dendogram* (Figure 2.3). The root of the dendogram contains all data points and leaf nodes represent individual data points. Internal nodes represents groups of adjacent data points. Each level of the dendogram presents a clustering of the dataset. A suitable clustering is chosen by selecting (cutting) an appropriate level of the dendogram. A sequence of clusterings of $\mathcal{D}$ can be obtained by two ways as follows.

1. **Divisive (Top-down) approach:** Divisive approach starts with a single cluster consisting of all patterns in it and subsequent clusterings are obtained by splitting clusters from previous clusterings. The idea of splitting is to disassociate a subset of patterns from others, which are not similar to the subset in a cluster. In other word, π_{i+1} is a refinement [Tremblay and Manohar (1997)] of π_i (Figure 2.3).

2. **Agglomerative (Bottom-up) approach :** Agglomerative approach starts with singleton clusters and each clustering π_i in the sequence is formed by merging

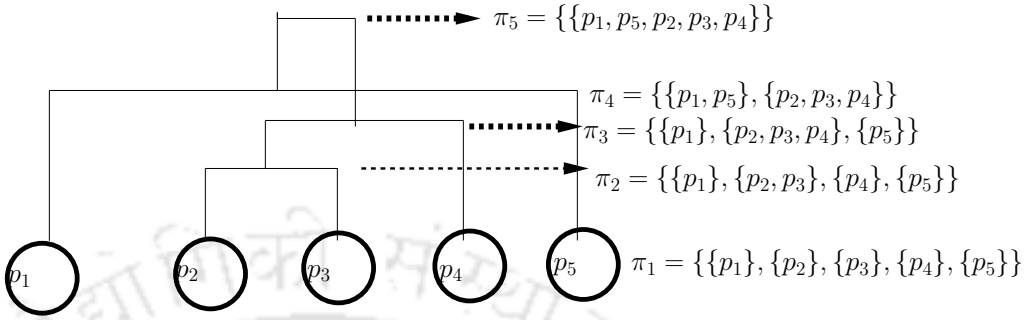


Figure 2.3: Dendogram produced by a hierarchical clustering method

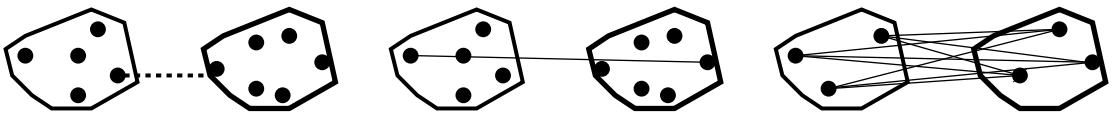
clusters of previous clustering π_{i-1}, where $i = 2..n-1$. In other word, π_i is a refinement of π_{i-1} (Figure 2.3).

A divisive clustering method [Gowda and Ravi (1995)] is proposed for symbolic dataset using *Minimum Spanning Tree* (MST). The method uses both similarity as well as dissimilarity for spliting a cluster into two sub clusters. Initially, all data points are in a cluster. A MST is generated for this cluster. It finds weakest edge (highest weighted edge) of MST and removes it to obtain two groups. Both similarity and dissimilarity are calculated between the two groups. If dissimilarity is more than similarity, then method proceeds to next iteration. This process is continued until no group has more dissimilarity than similarity. MONA, DIANA are other two divisive approaches [Kaufman and Rousseeuw (1990)].

Agglomerative approach is more popular than divisive approach. The Single-link [Sneath and Sokal (1973)], complete-link [King (1967)] and average-link [Murtagh (1984)] are examples of agglomerative hierarchical clustering methods. These three methods differ mainly in distance measures between a pair of clusters. Here, single-link method is discussed in detailed.

Single-link Clustering

Single-link [Sneath and Sokal (1973), Olson (1995), Berkhin (2006)] is a distance based clustering method which can find clusters of arbitrary (non-convexed) shapes and sizes in



(a) Single-link (b) Complete-link (c) Average-link

Figure 2.4: Distance between a pair of clusters in three hierarchal clustering methods.

data. Distance between a pair of clusters C_1 and C_2 is the minimum of distances between all pairs in $C_1 \times C_2$ (Figure 2.4(a)). That is,

$$Distance(C_1, C_2) = \min\{||x_i - x_j|| \mid x_i \in C_1, x_j \in C_2\}$$

Single-link method initially places each pattern in a separate cluster. The method calculates inter-cluster distances (called *distance matrix*) for the dataset scanning it multiple times. Iteratively, it merges a pair of closest clusters and updates the distance matrix accordingly. Iteration continues until all patterns belong to one cluster or a pre-specified condition is satisfied. Single-link outputs a dendogram. Whole procedure is given in Algorithm 2.2.

Disadvantages of single-link method are: (a) it scans a dataset multiple times, (b) Time and space requirements are $O(n^2)$. These are the serious drawback while working with the large dataset. (c) single-link suffers from chaining-effect– two clusters can be merged into one through noisy points.

Complete-link [King (1967)] overcomes the chaining problem considering the distance between two farthest patterns as the distance between a pair of clusters (Figure 2.4(b)). However, it is sensitive to noisy patterns. It produces globular shaped clusters.

Average-link [Murtagh (1984)] is less sensitive to noisy patterns. It is able to find clusters in the presence of noisy patterns. Distance between a pair of clusters C_1 and C_2 is the average distance between pairs of patterns, each of which is taken from separate cluster (Figure 2.4(c)). It also produces globular shaped clusters.

Lance and Williams [Lance and Williams (1967)] generalizes the distance measures (updation) between a cluster C_o and a new cluster $C = C_x \cup C_y$ (merging clusters C_x, C_y) for a number of hierarchical methods including single-link, complete-link, average-link

Table 2.1: Lance-Williams parameters for different hierarchal methods

Method	α_i	α_j	β	γ
Single-link	1/2	1/2	0	-1/2
Complete-link	1/2	1/2	0	1/2
Average-link	$\frac{m_{C_x}}{m_{C_x}+m_{C_y}}$	$\frac{m_{C_y}}{m_{C_x}+m_{C_y}}$	0	0
Centroid	$\frac{m_{C_x}}{m_{C_x}+m_{C_y}}$	$\frac{m_{C_y}}{m_{C_x}+m_{C_y}}$	$\frac{-m_{C_x}.m_{C_y}}{m_{C_x}+m_{C_y}}$	0

Algorithm 2.2 single-link($\mathcal{D}$)

Place each pattern $x \in \mathcal{D}$ in a separate cluster. This is the initial clustering $\pi_1 = \{C_1, C_2, \ldots, C_n\}$ of $\mathcal{D}$.
Compute the inter-cluster distance matrix and set $i = 1$.
while {Number of clusters in π_i is more than one or a per-specified constraint is satisfied} **do**
  Select two closest clusters C_x and C_y and merge into a single new cluster $C = C_x \cup C_y$.
  Next clustering $\pi_{i+1} = \pi_i \cup \{C\} \setminus \{C_x, C_y\}$.
  $i = i + 1$
  Update the distances from C to all other clusters using equation (2.12)
end while
Output the set of all clusterings $\pi_{\mathcal{D}} = \{\pi_1, \pi_2, \ldots, \pi_n\}$.

using equation (2.12).

$$d(C_o, (C_x, C_y)) = \alpha_i \times d(C_o, C_x) + \alpha_j \times d(C_o, C_y) + \beta \times d(C_i, C_j) + \gamma \times |d(C_o, C_i) - d(C_o, C_j)| \quad (2.12)$$

where, $d(.,.)$ is a distance function and values of $\alpha_i, \alpha_j, \beta$ and $\gamma (\in \mathbb{R})$ depend on the method used. For example, when $\alpha_i = \alpha_j = 1/2, \beta = 0$ and $\gamma = -1/2$, equation (2.12) reduces to equation (2.13), which is used in single-link method. Values of the parameters for all other hierarchical methods are shown in Table 2.1, where m_{C_x} and m_{C_y} are the number of patterns of clusters C_x and C_y, respectively. These all three methods have time and storage requirements of $O(n^2)$.

$$d(C_o, (C_x, C_y)) = \min\{d(C_o, C_x), d(C_o, C_y)\} \quad (2.13)$$

Single-link and complete-link clustering methods need to compute all pair pattern distances (distance matrix). Methods performs $n \times (n-1)/2$ distance computations, if distance function is a metric. To reduce the number of distance computations, Nanni [Nanni (2005)] exploits triangle inequality property of metric space to speed-up single-link and complete-link clustering methods. In this approach, triangle inequality is utilized to approximate distance between a pair of points as follows. Let x, y, z be three data points in $\mathcal{D}$ and d be a distance metric. Distances between x, y and y, z are $d(x, y)$ and $d(y, z)$, respectively. One can estimate distance between x and z denoted as $d^{EST}(x, z)$ without computing exact distance between them using equation (2.14)

$$|d(x, y) - d(y, z)| \leq d^{EST}(x, z) \leq d(x, y) + d(y, z) \tag{2.14}$$

Initially, a number of pivots p are selected from $\mathcal{D}$ and distances from each pivot to all other patterns in $\mathcal{D}$ are computed. Computation of all pair pattern distances is replaced by computing $p * |\mathcal{D}|$ exact distances and approximation of all other distances. The scheme exploits approximation distances to find two closest clusters in each iteration. Distance between two clusters are calculated by recursively analyzing their sub-clusters and applying equation (2.14) to them. In the analysis of sub-clusters, scheme always chooses sub-clusters with smaller lower bound distance to prune unnecessary distance computations. The scheme reported gain in the reduction of distance computations for a variant of single-link and complete-link methods (*k-cluster stopping condition*).

Koga et al. [Koga et al. (2007)] proposed a fast randomized approximation algorithm called *LSH-link*. LSH-link utilizes a probabilistic approximation algorithm called *Locality-Sensitive Hashing* (LSH) [Indyk and Motwani (1998)], which is used for finding nearest neighbors of a query point. The LSH-link has multiple phases. In each phase, algorithm quickly finds neighbor clusters of each cluster using LSH and merges them. A cluster C_1 is considered to be neighbor of another cluster C if distance between them is less than r. The distance parameter r is pre-specified and changes in each phase. Each phase corresponds to a clustering or a layer in the dendogram. The algorithm terminates if a phase has only one cluster. The clustering results produced by LSH-link is same (close) as the results

produced by classical single-link method. It is shown that LSH-link runs in linear time under certain assumptions. It can find clusters of arbitrary shapes and sizes. However, clustering results are highly influenced by parameters (number of hash functions, number of entries in hash tables, initial threshold distance, ratio of two successive layer-distances of dendogram). For high dimensional dataset, size of hash table could be very large and unmanageable.

Dash et al. [Dash et al. (2003)] proposed a fast hierarchical clustering method, which is based on partially overlapping partition (POP). In POP, entire feature space is partitioned into a number of overlapping cells. Width of overlapping region of a pair of cells is δ. POP exploits empirical observation named *90-10 rule* to avoid the computation of all inter-pattern distances. The rule states that clusters at lower levels of a dendogram are very small in size and they are separated by very small distance (a small fraction of maximum closest pair distance). Therefore, POP examines each cell individually as follows. Closest pair-distance is calculated for each cell and if overall closest pair-distance is less than δ, then the pair is merged and distance update is performed in the concerned cell only and all other cell remain unaffected. In this way, scheme avoids unnecessary distance computations from newly merged cluster to clusters in other cells. This continues until overall closest pair distance is more than δ. In next phase, traditional hierarchical agglomerative clustering (HAC) method is applied to the clusters obtained in first phase. Here, HAC uses centroid based metric to calculate distance between clusters.

However, all these clustering methods keep entire dataset in main memory for processing. This prohibits its application to large dataset. BIRCH [Zhang et al. (1996)] is hierarchical clustering method which is designed to work in a limited main memory and has linear I/O time (single database scan). Therefore, BIRCH is suitable for clustering large dataset.

Several clustering methods have been proposed to combine partitional clustering method with hierarchical clustering method to get advantages of both the methods [Murty and Krishna (1981), Wong (1982), Lin and Chen (2005), Liu et al. (2009)]. This type of clustering

method is known as *hybrid clustering method*. Main objective of this approach is to reduce the time and space requirements of hierarchical method keeping clustering quality almost same as that of the hierarchical methods.

2.3.4 Hybrid Clustering

First hybrid method was proposed by Murthy and Krishna [Murty and Krishna (1981)]. It is based on multilevel approach [Murty and Krishna (1980)]. In the first level, dataset $\mathcal{D}$ is divided into a number of subsets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_p$ such that $\bigcup_i \mathcal{D}_i = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, i \neq j$. Next, k-means clustering method is applied to each $\mathcal{D}_i$ to obtain C_1 sub-clusters. Centroids of all sub-clusters of $\mathcal{D}$ forms a set of centroids, which is used in the next level. In this level, set of centroids of size $C_1 \times p$ is divided into a number of sub-sets and single-link method is applied to each sub-set of centroids for merging of adjacent centroids. Operations (division of centroids and merging them using single-link) of previous level are repeated until final level produces pre-specified number of clusters. Finally, centroids in each level are replaced by original data points of the corresponding clusters. This method can find non-convexed shaped clusters in data. Clustering results are empirically analyzed only. It needs many parameters which are difficult to determine in real world dataset. Similarity between a pair of centroids may not reflect actual similarity between the corresponding clusters, specially when cluster size is large.

Hybrid clustering method proposed by Wong [Wong (1982)] is a combination of k-means and single-link method. It has two stages. In the first stage, k-means is applied to estimate density over k clusters of entire dataset. Here, distance between a pair of clusters is a function of the estimated densities of the clusters. Wong showed that hybrid method can identify high density clusters (set-consistence) in one dimension data as that of the single-link clustering [Hartigan (1981)]. However, there is no generalized analysis for more than one dimensions.

In the same line other hybrid clustering methods are proposed in [Lin and Chen (2005), Liu et al. (2009), Chaoji et al. (2009)], which divide dataset into a number of

sub-clusters applying k-means method and subsequently an agglomerative method is applied to sub-clusters for final clustering. These methods differ in measuring similarity (separation) between a pair of sub-clusters. CSM [Lin and Chen (2005)] measures cohesion (inter-cluster distance) between a pair of sub-clusters. Cohesion between a pair of sub-clusters is measured using probability density function (pdf) at each point of a sub-cluster. However, probability density function is assumed to be followed normal distribution. M. Liu et al. [Lin and Chen (2005)] proposed to use multiple prototypes as representatives for a cluster. In this approach, k-mean is used to produce a large number of prototypes to locate high density regions. A separation (distance) measure is introduced to quantify separateness (closeness) between a pair of sub-clusters. SPARCL [Chaoji et al. (2008), Chaoji et al. (2009)] also introduces a novel similarity measure, which is suitable for finding arbitrary shaped clusters in dataset. Here, similarity function is proportional to density of the sub-clusters and inversely proportional to distance between them. Since, size of similarity matrix is less, spectral clustering [Ng et al. (2001)] can also be applied in the second stage of SPARCL method.

Vijaya et al. [Vijaya et al. (2006)] proposed a hybrid clustering method to speed up protein sequence classification. It is a two-stage algorithm. In the first stage, leaders clustering is applied to dataset for finding a set of sub-cluster representatives (leaders). In next stage, a complete-link/single-link clustering method is applied to the leaders to obtain k clusters. A median of each of k clusters is selected as cluster representative. To classify a test pattern, a classifier selects the closest cluster representative and subsequently closest sub-cluster in that cluster. Finally, it is classified with the class label of nearest of the two representatives. In this method, the hybrid clustering method is used to reduce computational cost of the classifier only. However, no theoretical and experimental analysis of clustering results are reported in the paper. Relation between leaders threshold (τ) and the number of cluster representatives (k) are also not reported.

The thesis has made a contribution in developing a hybrid clustering method for finding arbitrary shaped clusters in large dataset.

2.3.5 Density based clustering method

Density based clustering method views clusters as dense regions in the feature space which are separated by relatively less dense regions. Basic approach of density based clustering method is to find density information of each pattern in the dataset. A cluster is formed by merging (connecting) nearby patterns with almost equal density. Main advantage of density based method is that it can find clusters of arbitrary shapes and sizes along with outlier points.

DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [Ester et al. (1996)] is a very popular density based partitional clustering method. Idea of DBSCAN is based on concept of densed points and connectivity relation between them. A point $x \in \mathcal{D}$ is said to be dense in the feature space if a hyper-sphere of radius ϵ ($\in \mathbb{R}^+$) centered at x has at least $Minpts$ ($\in \mathbb{N}$) numbers of points.

A pair of points x and y is said to be connected if any one of the conditions is satisfied.

1. Either x or y is a dense point and $||x - y|| \leq \epsilon$
2. There is a sequence of patterns $x_1 = x, x_2, \ldots, x_{m>1} = y$ such that $||x_i - x_{i+1}|| \leq \epsilon$ and all x_i, $i = 1..(m-1)$ are dense points or vice-versa (*density reachable* (Figure 2.5)).
3. There is a dense point x_c so that there exist two sequences of points $x_1 = x_c, x_2, \ldots, x_{m_1} = x$ and $y_1 = x_c, \ldots, y_{m_2} = y$ such that $||x_i - x_{i+1}|| \leq \epsilon, ||y_j - y_{j+1}|| \leq \epsilon, x_i, i = 1..(m_1 - 1)$, and $y_j, j = 1..(m_2 - 1)$ are dense points (*density connected* (Figure 2.5)).

A cluster $C \subset \mathcal{D}$ holds following properties.

- $\forall x, y \in C$, x and y are density connected.
- If $x \in C$ is a dense point, all density-reachable points from x are included in C.

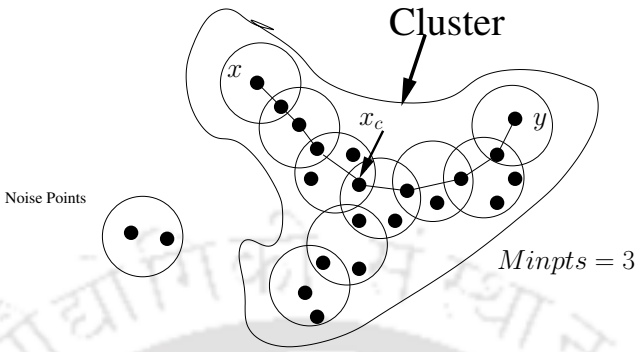


Figure 2.5: Cluster produced by DBSCAN:Points x and y are density connected through x_c; Point x is density reachable from x_c.

DBSCAN visits each pattern in a dataset. If it finds a dense point, it starts expanding a cluster recursively by including all density reachable points to the cluster. Finally, points which do not include to any cluster are treated as noisy/outlier points. The complexity of DBSCAN is $O(n^2)$ without using any spatial indexing scheme. Drawbacks of DBSCAN are as follow.

i) DBSCAN is sensitive to its parameters. Determining values of parameters ϵ, $Minpts$ are difficult task for real world large dataset.

ii) DBSCAN fails to detect clusters of arbitrary shapes with highly variable density dataset.

iii) DBSCAN is not scalable with the size of the dataset.

OPTICS (Ordering Points To Identify the Clustering Structure)

OPTICS [Ankerst et al. (1999)] is a density based method. OPTICS creates an augmented *ordering* of points in dataset, which represents hierarchical clustering structures of the data. It works with two parameters ϵ, $Minpts$ as that of DBSCAN. However, OPTICS produces different clusterings (partitions) simultaneously with different values of $\epsilon'(<= \epsilon)$ and a fixed value of $Minpts$. OPTICS performs range queries for each data point in the dataset. However, spatial indexing method does not perform well in high

dimensional dataset. As a result, OPTICS does not perform well in high dimensional large dataset. Determining the value of ϵ, $Minpts$ are difficult task.

DenClue

DenClue [Hinneburg and Keim (1998)] is a density based clustering method. Unlike DBSCAN, DenClue computes density function, which is the summation of *influence functions* in the feature space. The influence function describes the impact of a data point within its neighborhood region. Various influence functions are discussed to generalize different clustering algorithms (DBSCAN, k-means, hierarchical methods). Clusters are identified by finding local maxima of density functions using gradient hill-climbing technique. Clusters of arbitrary shapes are defined as sequence of data points with local (maxima) density more than pre-specified threshold. However, DenClue requires many parameters which need to be adjusted for proper clustering output. Worst case time complexity of DenClue is $O(n^2)$, where n is the size of the dataset. DENCLUE 2.0 [Hinneburg and Gabriel (2007)] speeds up DenClue method by introducing a new hill climbing procedure for Gaussian influence function.

Chameleon [Karypis et al. (1999)]

Chameleon is an agglomerative hierarchical clustering method. It utilizes a popular graph partitioning approach [Karypis and Kumar (1998)] in first step to partition K-neighbor graph of a given dataset. This step produces a number of sub clusters. In the next step an agglomerative approach is applied to these sub clusters to obtain final clusters of the dataset. It uses a dynamic modeling in cluster aggregation. Two sub clusters are merged in the aggregation process only if the inter-connectivity and closeness between two sub clusters are high relative to the internal inter-connectivity and closeness of items within the sub clusters. Chameleon outperforms other hierarchical clustering method like CURE [Guha et al. (2001)]. This method can find clusters of different shapes, densities, and sizes in two-dimensional space. It needs many parameters and has time complexity of $O(n^2)$.

NBC (Neighborhood Based Clustering Algorithm)

NBC is a density based [Zhou et al. (2005)] clustering method which can find clusters of arbitrary shapes, different sizes and distributions. NBC calculates local density of each data point by introducing a new measurement of density called *Neighborhood Based Density Factor (NDF)*. The NDF of a point is ratio of the number of reverse K-nearest neighbors to the number of K-nearest neighbors. Reverse K-nearest neighbors of a point p is the set of patterns whose K-nearest neighbors contain p. For most of the patterns, number of K-nearest neighbors is K, (Some cases, it may be more than K, but not less than K). However, value of reverse K- nearest neighbor may have less than or more than K based on the position of the point in the feature space. Based on the value of NDF, a point can be categorized into any one of the three types: *dense point, sparse point and even point.* If $NDF(p) > 1$, then point p is called dense point. Larger the $NDF(p)$ is, denser (inlier-ness) the point is. If $NDF(p) < 1$, then point p is called sparse point. If $NDF(p) = 1$, then point p is called even point. Using these information, NBC method defines cluster of a dataset in the same line of DBSCAN as follows.

Let C be a subset of $\mathcal{D}$. The C is called a cluster if it satisfies following properties:

i) $\forall x, y \in C$, there is a sequence of objects $x_1 = x, x_2, \ldots, x_m = y$ such that x_{i+1} is a K-nearest neighbor object of x_i, for $i = 1 \ldots (m-1)$ and each x_i is either dense or even point.

   OR,

   There exist two sequences of points $x_1 = x_c, x_2, \ldots, x_{m_1} = x$ and $y_1 = x_c, y_2 \ldots, y_{m_2} = y$ such that x_{i+1}, $y_{j+1} \in K$-nearest neighbor of x_i and y_j, respectively, and each $x_i, i = 1..(m_1 - 1)$, and each $y_j, j = 1..(m_2 - 1)$ is either dense or even point.

ii) If x is a dense/even point of C, then C includes points of all sequences $x_1 = x, x_2, \ldots, x_l$ for any positive integer of l such that each $x_i, i = 2..(l-1)$ is either dense or even point.

NBC has two steps. In the first step, it calculates NDF of each point in the dataset. In the second step, it finds clusters by checking NDF value of each point in the dataset.

If a point p is dense or even point, a new cluster with p is created. NBC then iteratively adds K -neighbors points to this cluster by checking NDF values. This process stops when there is no dense or even points. It may be mentioned that it simultaneously finds outliers in the dataset. NBC uses only one parameter K to find clusters. However, with a slight change in value of K, it produces different clustering results. Genuine clusters are treated as outlier points as NBC only use K-neighbors and reverse K-neighbors counts. It does not use relative distance (position) of a point in its neighborhood region.

A density based method which outperforms NBC method is proposed as a contribution in the thesis.

2.3.6 Data Summarization

One way to handle large dataset is to use data summarization or data compression technique. The main objective of this approach is to compress a large amount of dataset into a small representative set called summarized set. Then, classical clustering method is applied to this set. Finally, representatives are explored by the original data points in order to obtain clustering of the whole dataset.

BIRCH [Zhang et al. (1996)] is an incremental clustering method. It has two phases. In the first phase, it create a summary of the dataset, which can be accommodated in the available main memory of the machine. In the second phase, conventional hierarchical clustering method such as average-link is applied to the summary for obtaining final clustering of the entire dataset.

The core idea of the BIRCH clustering method are *Clustering Feature (CF)* and *CF-tree*. A CF is a triplet which contains summarized information of a sub-cluster. More formally, a CF for a sub-cluster $C_1 = \{\overrightarrow{X_1}, \overrightarrow{X_2}, \ldots \overrightarrow{X_k}\}$ is defined as $CF = (k, \overrightarrow{LS}, ss)$, where $\overrightarrow{LS}$ is linear sum of patterns in C_1, i.e., $\overrightarrow{LS} = \sum_i \overrightarrow{X_i}$, ss is square sum of data points, i.e., $ss = \sum_i \overrightarrow{X_i}^2$. Centroid, radius and diameter of sub-cluster C_1 can be computed from the CF values. CF-values satisfy additivity property, i.e. let $CF_1 = (k_1, \overrightarrow{LS_1}, ss_1)$ and $CF_2 = (k_2, \overrightarrow{LS_2}, ss_2)$ are two CF values of two sub-clusters C_1 and C_2, respectively. Then,

CF-value of $C = C_1 \cup C_2$ is $CF(C) = (k_1 + k_2, \overrightarrow{LS_1} + \overrightarrow{LS_2}, ss_1 + ss_2)$. CF values of sub-clusters are organized incrementally in a CF tree, which is a height-balanced tree. Each internal node of CF tree contains at most B entries of the form $[CF_i, child_i]_{i=1,\ldots,B}$, where CF_i is the cluster feature of sub-cluster represented by this child and $child_i$ is a link to i^{th} child. A leaf node has at most L_{max} entries of the form $[CF_i], i = 1 \ldots L_{max}$. However, diameter of each sub-cluster in leaf nodes must be less than a threshold τ. Diameter of a sub-cluster means the average of the pairwise distance between all points in the sub-cluster. To insert a new pattern into CF tree, scheme starts searching nodes from root to the leaf selecting closest node at each level. Distance is measured from new pattern to the centroid of sub-cluster. Once new point is absorbed in a leaf node, the CF values of each node on the traversed path from root to the leaf is updated applying the additive property. Node split may occur if there is no space in a given node. Spliting is performed in the same line as performed in a B-tree. Leaf entries represent summary of the whole dataset. Finally, a classical clustering method such as average-link method is applied to the leaf-entries of the CF tree. BIRCH scans the dataset only once and more scans refine clustering output. Time complexity is linear with the size of the dataset. Main drawbacks of this approach are that (a) it cannot detect arbitrary shaped clusters, (b) it needs many parameters-B, L_{max}, τ, (c) it is order dependent method, (d) BIRCH is applicable to numeric data only and feature space must satisfy *vector-space* properties (vector addition and scalar multiplication). CF-values are utilized to speed up different partitional (like k-means, EM) and hierarchical (like OPTICS) clustering methods [Bradley et al. (1998), Breunig et al. (2000, 2001)].

Bradley [Bradley et al. (1998)] utilizes clustering feature to speed up k-means clustering method for large dataset. The method can be run in available main memory of a machine. The key insight of this approach is that all data points are not equally important to a cluster. Therefore, method categorizes each data point into one of the three categories namely *retain set* (RS), *discard set* (DS) and *compressed set* (CS) based on the position of the point in a cluster. A DS set corresponding to a cluster is a compressed representation of points which lie within a Mahalanobis radius r of a hyper-sphere centered

at centroid of the cluster. The hyper-sphere covers ρ % of data points assigned to the cluster. These points do not change cluster means in future iterations. Statistics of these points are stored in CF triplet and original points are purged from the main memory. The data points which cannot be part of DS set form CS set or RS set. These points change their cluster membership. A variant of k-means clustering is applied to these points with a large number of clusters $k_2 > k$. Having obtained the k_2 sub clusters, points of each sub cluster close to center are compressed in CF form called CS set. Here, closeness of a point to a center is determined from covariance value of the sub cluster, which can easily be calculated from CF value of the sub cluster. The k_2 sub clusters are merged with existing sub clusters in CS set using an agglomerative clustering method. RS set contains points which fail closeness test of a sub cluster and points in RS are retained in the memory. Finally, compressed statistics and original data points are used to speed up many clustering methods in large dataset.

DuMouchel et al.[DuMouchel et al. (1999)] also proposed a data summarization technique, which can be used to scale up a set of machine learning and statistical modeling methods. The method is called *data squashing*. Data squashing summarizes a large dataset into smaller size dataset called *squashed data* with same number of dimensions as the original dataset. It has three sequential steps as follow. In the first step, dataset is partitioned into a number of compact sub regions or bins. Bins can be constructed by creating hyper-rectangles in the original space or by creating different *layers* in the transformed data space. Let $p = (x_1, \ldots, x_N)$ be a point in the original space. The point p can be transformed into $p' = (y_1, \ldots, y_N)$, where $y_i = \dfrac{C_i - x_i}{\text{standard deviation of } i^{th} \text{ feature}}$, $C_i = i^{th}$ component of center $C = \dfrac{\sum_{\mathcal{D}} p}{|\mathcal{D}|}$ of the whole dataset. Layers are created based on the distance from p' to origin. Having created the bins, next step computes moments for the data points falling in a bin. Method computes different order moments like means, minima, maxima, second order moments, third order moments, fourth order moments, etc. in each bin. These moments are sufficient statistics of the points of a bin. Last step is to create a pseudo data point corresponding to a bin from the above calculated statistics. A pseudo point of a bin is the representative of all original data points falling

in the bin. They showed that data squashing outperform random sampling approach. These data summarization schemes cannot compute effective distance between a pair of summarized units. Therefore, these schemes cannot be directly applied to hierarchical clustering methods.

First successful data compression technique for hierarchical clustering method is found in [Breunig et al. (2000)]. The compressed representation of a set of data points are called *Data Bubble (DB)*, which contains statistical information of the set of points. Data Bubble initially selects a subset of patterns randomly which requires one database scan. In next step, it classifies each pattern to its nearest selected pattern and updates statistics of the selected pattern by means of CF. Data Bubble utilizes Clustering Features (CF) introduced in BIRCH clustering method as follows. Let $CF = (k, \overrightarrow{LS}, ss)$ be the Clustering Features of a set of points $X = \{X_i\}$ in N dimensions feature space. A Data Bubble B, which describes X can be expressed as $B = (k, \overrightarrow{M}, e)$, where $\overrightarrow{M} = \dfrac{\overrightarrow{LS}}{k}$ is the center of X and $e = \dfrac{\sqrt{2 * n * ss - 2 * \overrightarrow{LS}^2}}{k * (k-1)}$ is called extent (spread) of X. More specifically, e is the radius of a hyper-sphere centered at M, which covers most of the data points of X. Markus et al. [Breunig et al. (2000)] proposed to apply OPTICS method to the data bubbles of $\mathcal{D}$. OPTICS clustering method always finds an unprocessed closest data points from already processed item. However, distance between centers of a pair of Data Bubbles does not reflect the distance between points of the Data Bubbles. A suitable measure is introduced to calculate distance between a pair of Data Bubbles. Finally, OPTICS clustering method is modified to work with Data Bubbles. Modified OPTICS is found to be up to three order faster than classical OPTICS method applied directly to dataset with maintaining clustering quality.

In the following year, Markus et al. [Breunig et al. (2001)] pointed out problems of applying OPTICS method directly to Clustering Features (CF). Clustering quality deteriorates abruptly with high compression ratio due to three key problems namely *size distortion*, *lost objects* and *structural distortion*. Sizes of clusters are distorted with high compression ratio (number of representative objects to the size of the dataset), *i.e.*, some clusters become larger and other become smaller. This is called "size distortion" problem.

OPTICS outputs a *reachability-plot,* which is a two dimensional plot represents hierarchal structures of clusters present in the dataset. None of the original objects appears in the reachability plot if CF is directly used in speeding up OPTICS method. This is called "lost objects" problem. With high compression rate, clustering structures are distorted in the reachability plot, this is called "structural distortions". To alleviate all above problems, definition of Data Bubble is redefined as follows.

Definition 2.1 (Data Bubble) *[Breunig et al. (2001)] A Data Bubble for a set of points* $X = \{X_i\} \subseteq \mathcal{D}$ *is* $B = (rep, k, extent, nnDist)$, where

- *rep* is representative object of X
- k is the number of objects in X
- *extent* is the radius of hyper-sphere centered at *rep* which encloses most of the objects in X.
- *nnDist(K,B)* is a function that calculate average K-nearest neighbor distances in B.

Distance between a pair of Data Bubbles is introduced, which is suitable for hierarchical clustering method. Distance is measured in such a way that it approximately measures distance between two closest points of the Data Bubbles. Breunig et al. showed that data bubble outperforms BRICH [Zhang et al. (1996)] as a summarization scheme when the OPTICS method is applied to it. The main disadvantage of Data Bubble approach is that it assumes that dataset is from vector space. The approach may underestimate effective distance between a pair of bubbles in some cases. For example, let two data bubbles (O_1, O_2) are close (neighbor) to each other and one data bubble (say O_1) contains patterns from two different clusters, then cluster separation ("gap") becomes invisible to data bubble O_1 (Figure 2.6). Therefore, effective distance between two data bubbles is underestimated and bubbles get merged. This leads to errors in final clustering results.

In order to overcome both the drawbacks of the Data Bubbles, Zhou and Sander [Zhou and Sander (2003)] introduced "directional" notion to the data bubble, which

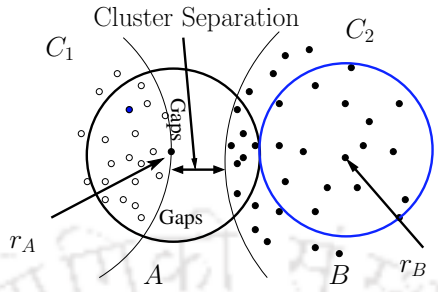


Figure 2.6: Data bubble A has "gap" in the direction of data bubble B

does not use any vector-space properties (vector addition and scale multiplication) unlike previous approaches. This "directional" Data Bubble uses only similarity or dissimilarity measure. In this approach, relative position of each point $x \in O_1$ is located by two ways - whether x is in the direction of data bubble O_2 or in the reverse direction of O_2. The point $x \in O_1$ is in the direction of O_2, if $||x - r_{O_2}|| \leq ||r_{O_1} - r_{O_2}||$, where r_{O_1} and r_{O_2} are the representative patterns of Data Bubbles O_1 and O_2, respectively. Otherwise, x is in the reverse direction (*i.e.* $||x - r_{O_2}|| > ||r_{O_1} - r_{O_2}||$). In order to compute accurate distance between a pair of Data Bubbles the concepts of "border distance", "extent" of a Data Bubble are introduced in this work. More formally, let O_1 and O_2 be two "directional" Data Bubbles with representatives r_{O_1} and r_{O_2}, respectively. Border distance of O_1 in the direction of O_2 is $Border(O_1)^{in.O_2}$, which is defined as follows.

$$Border(O_1)^{in.O_2} = ||r_{O_1} - r_{O_2}|| - \min_{x \in O_1}\{||x - r_{O_2}||\}$$

Border distance of O_1 in the direction of O_2 is the difference of distances between two representatives of O_1 and O_2, and between representative of O_2 and the object $x \in O_1$ closest to it. However, "gap" may make border distance overestimated in the direction in which the "gap" exists in a bubble. To avoid this problem, notion of "directional" border distance called "directional extent" or *extent* is introduced. The extent of a Data Bubble O_1 in the direction of other data bubble O_2 is the function of border distance of O_1, average and standard deviation of distances of objects in O_1 in the both directions of

O_2, i.e.,

$extent(O_1)^{O_2} = \min\{Border(O_1)^{in.O_2}, avg + 2 * stdv\}$,

where avg and $stdv$ are the minimum of the average and standard deviation of the distances of objects in O_1 in the direction and the reverse direction of O_2. Average (d_{avg}) and standard deviation (d_σ) of a Data Bubbles in the direction of other data bubble are calculated as follows. Consider a Data bubble O_1 has k objects in the directions of another data bubble O_2. Then, $d_{avg} = \frac{lsD}{k}$; $d_\sigma = \sqrt{\frac{k * ssD - (lsD)^2}{k^2}}$, where lsD and ssD are the Linear sum and square sum of distances between representative of O_1 and points in O_1, which can be collected incrementally. Finally, distance between two data bubbles O_1 and O_2 can be measured using equation (2.15).

$$dist(O_1, O_2) = ||r_{O_1} - r_{O_2}|| - extent(O_1)^{O_2} - extent(O_2)^{O_1} \quad (2.15)$$

Distance between two Data bubbles is the distance between their representatives minus extents of the bubbles with respect to each other. This approach of data bubble outperforms other approaches of data bubble [Breunig et al. (2000, 2001)] when the OPTICS method is applied to it.

Soft computing techniques may play major role in deciding the actual membership of border points of Data Bubbles. In next, we discuss some soft computing approaches. Thesis has made a contribution in developing a summarization scheme, which uses soft computing technique.

2.3.7 Soft Clustering

In many application domains like web mining, document clustering, bio-informatics clusters do not have crisp (exact) boundary. They overlap each other in the feature space. Traditional clustering methods produce non-overlapped (disjoint) clusters in which a pattern belongs to single cluster, called *hard cluster*. Overlapped clusters are called *soft* clusters and corresponding clustering is called *soft clustering*. In soft clustering, a pattern

may belong to more than one clusters. Soft clustering has strong mathematical foundations -*fuzzy set theory* [Zadeh (1965)], *rough set theory* [Pawlak (1982)] and hybridization of both approaches.

Fuzzy Set Theory and Clustering Method

Fuzzy set theory was developed by L. A. Zadeh to handle incomplete, ambiguous information in pattern recognition and machine learning domains [Zadeh (1965)]. Let $\mathcal{D} = \{x_i\}$ be a set of patterns. A fuzzy set (class) A associated with $\mathcal{D}$ is described by a membership function $f_A : \mathcal{D} \rightarrow [0, 1]$, which quantify the belonging-ness of each pattern $x \in \mathcal{D}$ in A. Closer the value of f_A to 1, higher the grade of membership.

Fuzzy set theory has wide applications in industry like process control systems since late 1980s. Success of fuzzy systems inspired to use it in information systems. First known fuzzy clustering method was proposed by Dun [Dun (1973)]. The proposed method is known as Fuzzy c-Means Method (FCM), which was generalized by many researchers later on [Gustafson and Kessel (1979), Bezdek (1981), Hathaway and Bezdek (1988)].

FCM is based on the concept of *fuzzy c-partition* [Ruspini (1969)]. A fuzzy c-partition P_{fc} of a dataset $\mathcal{D}$ of size n can be represented as follows.

$$P_{fc} = [u_{ik}]_{c\times n} : u_{ik} \in [0, 1], \text{ where } \sum_{i=1}^{c} u_{ik} = 1 \text{ and } 0 < \sum_{k=1}^{n} u_{ik} < n \tag{2.16}$$

where u_{ik} is the membership value of pattern $x_k \in \mathcal{D}$ to cluster C_i. FCM optimizes the following objective function in order to obtain a fuzzy c-partitions.

$$J_{FCM}(C, V) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \ ||x_k - v_i||_2{}^2, \tag{2.17}$$

where $v_i = \dfrac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m}, m \in [0, \infty]$.

FCM starts with c random patterns considering them initial centroids V. It calculates fuzzy membership value u_{ik} of each pattern x_k, which gives initial fuzzy c partition.

Membership value u_{ik} must satisfy the constrains given in equation (2.16) and it is the function of distance from a pattern x_k to all other cluster centroids. It iteratively updates centroids V and membership values of each pattern until gets converged. Main drawback of this is finding proper parameters and computationally more expensive than classical k-means method.

Other complementary approach Rough Set Theory is used to capture uncertainties associated with data. In the next, we provide brief introduction to Rough Set Theory and well know rough clustering methods are discussed.

Rough Set Theory and Clustering Method

Rough set theory was introduced by Polish mathematician Pawlak [Pawlak (1982), Lin and Cercone (1996)]. The fundamental idea of the rough set theory is based on an approximation space $\mathcal{A} = (U, R)$, where U is a nonempty set of objects and R is an equivalence relation (reflexive, symmetric, and transitive) called indiscernibility relation on U [Pawlak (1982)]. R creates a partition U/R of U, *i.e.*

$$U/R = \{X_1, \ldots, X_i, \ldots, X_p\}$$

where each X_i is an equivalence class of R. These equivalence classes and the empty set are considered as the Elementary sets in $\mathcal{A}$. Elementary sets form the basic granules of knowledge about any set of objects U. Any arbitrary set $X \subseteq U$ can be defined by two crisp sets called lower and upper approximation of X. Each crisp set is finite union of the Elementary sets in $\mathcal{A}$. More formally, one can define the lower and upper approximation as follow (Figure 2.7).

$$\underline{R}(X) = \bigcup_{X_i \subseteq X} X_i; \quad \overline{R}(X) = \bigcup_{X_i \cap X \neq \emptyset} X_i.$$

The set $BND(X) = \overline{R}(X) - \underline{R}(X)$ is called *boundary* of X in $\mathcal{A}$. Sets $\underline{Edg}(X) = X - \underline{R}(X)$ and $\overline{Edg}(X) = \overline{R}(X) - X$ are called *internal* and *external* edge of X in $\mathcal{A}$, respectively [Pawlak (1982)].

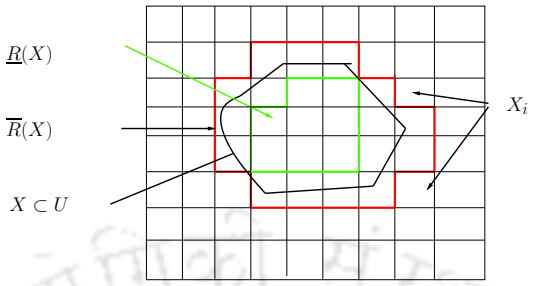


Figure 2.7: Lower and Upper Approximations of X in Rough Set Theory

Rough k-means is an adapted version of k-means clustering method in rough set theory [Lingras and West (2004)]. It can produce overlapped clusters in many applications. Rough k-means is based on the idea of lower and upper approximation of a subset of $\mathcal{D}$. The rough k-means modifies a centroid C_i of a cluster $C \subset \mathcal{D}$ as follows.

$$C_i = \begin{cases} w_{lower} \times \dfrac{\sum_{x_i \in \underline{R}(C)} x_i}{|\underline{R}(C)|} + w_{upper} \times \dfrac{\sum_{x_i \in \overline{R}(C) - \underline{R}(C)} x_i}{|\overline{R}(C) - \underline{R}(C)|} & \text{if } \overline{R}(C) - \underline{R}(C) \neq \emptyset \\ w_{lower} \times \dfrac{\sum_{x_i \in \underline{R}(C)} x_i}{|\underline{R}(C)|} & \text{otherwise} \end{cases} \tag{2.18}$$

where w_{lower} and w_{upper} are two parameters, which correspond to relative importance of lower and upper approximation. It may be noted that equation (2.18) reduces to equation (2.19), when there is no point in upper approximation of the cluster, i.e. rough boundary is empty.

$$C_i = \frac{\sum_{x_i \in \underline{R}(C)} x_i}{|\underline{R}(C)|}, \text{ where } \underline{R}(C) = C \tag{2.19}$$

In each iteration, rough k-means examines each object x and assigns it to a lower approximation of a cluster or upper approximations of more than one clusters as follow.

- If C_i is the closest centroid and there are centroids C_j such that $d(x, C_i) - d(x, C_j) \leq \tau_1$, then x is assigned to upper approximations of all C_j and C_i centroids, i.e. $x \in \{\overline{R}(C_i), \overline{R}(C_j)\}$. The pattern x will not be part of lower approximation of any cluster.

- Let C_i be the closest centroid and C_j be the second closest to x. Then, pattern x belongs to only lower approximation of C_i, if $d(x, C_j) > \tau_1 + d(x, C_i)$.

The main disadvantage of this method is the requirements of many user defined parameters $(w_{lower}, w_{upper}, \tau_1)$, which is a difficult task to estimate in real dataset.

Tolerance Rough Set Model

The core idea of rough set theory is the indiscernibility relation, which is an equivalence relation on U. An equivalence relation R must be reflexive, symmetric, and transitive. However, it is observed that the transitive property does not hold in certain application domains (*i.e.* document clustering, information retrieval [Ho and Nguyen (2002)]), which leads to emerge a generalized rough set theory called Tolerance Rough Set Model (TRSM) [Nie, Polkowski et al. (1995), Skowron and Stepaniuk (1996), Kryszkiewicz (1998)], [Ślezak and Wasilewski (2007)].

Basic granules of knowledge in TRSM is tolerance classes, which are overlapped, intermixed unlike equivalence class in Rough Set. Therefore, tolerance relation T does not create a partition of U. It creates a cover of the dataset. Any set $X \subseteq U$, can be characterized by the lower and upper approximations as follow (Figure 2.8).

$$\underline{T}(X) = \{x \in X : T(x) \subseteq X\}; \quad \overline{T}(X) = \{x \in U : T(x) \cap X \neq \emptyset\}$$

where $T(x)$ is a tolerance class. In accordance with the classical rough set theory, one can define the set $T_{BND} = \overline{T}(X) - \underline{T}(X)$ as *tolerant boundary*. The sets $T_{\underline{Edg}} = X - \underline{T}(X)$ and $T_{\overline{Edg}} = \overline{T}(X) - X$ are termed as *tolerant internal* and *tolerant external* edge of X, respectively (Figure 2.8). R. Slowinski and Vanderpooten argued in favor of not imposing the symmetric property along with transitive property in generalized rough set [Slowinski and Vanderpooten (2000)]. A relation T is said to be symmetric on U,

$$\forall a, b \in U, \ aTb => \ bTa$$

Say a and b are related to both way (direction), i.e. a is related to b and b is related to a.

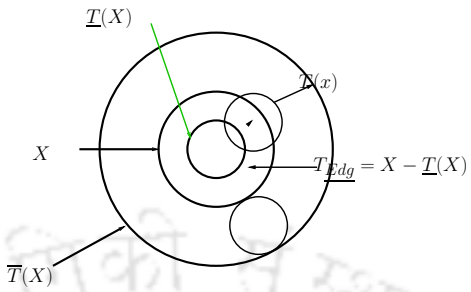


Figure 2.8: Lower and Upper Approximation of X in TRSM

A new relation called *similarity* is defined, which maintains a certain form of *symmetric* relation, i.e. *inverse relation* T^{-1}. In the same line of RST and TRSM, similarity class and inverse class of $x \in U$ are defined as follow.

$$T(x) = \{y \in U : yTx\} \qquad T^{-1}(x) = \{y \in U : xTy\} \tag{2.20}$$

Finally, lower and upper approximation of a subset of U are defined and compared with other approach of rough set theory. Other direction of advancement of classical rough set theory is found in [Bedi and Chawla (2009)].

This thesis uses TRSM approach to handle the uncertainty of leaders clustering method. Euclidean distance measure is used in leaders method and it is symmetric distance. Therefore, tolerance relation is most appropriate.

T. Ho *et al.* [Ho and Nguyen (2002)] proposed a tolerance rough set model (TRSM) based document clustering. In this approach, tolerance relation and tolerance class are defined based on co-occurrence of index terms present in document set. Co-occurrence of words gives meaningful interpretation and semantic relation between them. Let t_i be an index term and $f_{\mathcal{D}}(t_i, t_j)$ be the number of documents in which t_i, t_j co-occur. The tolerance class of t_i is defined as

$$T(t_i) = \{t_j \in \mathcal{T} \mid f_{\mathcal{D}}(t_i, t_j) \geq c_2\},$$

where $\mathcal{T}$ is the set of index terms. Similarity between a pair of documents is the function

of upper approximations of the index terms of the documents. Upper approximation of $X \subseteq \mathcal{T}$ is $\overline{T}(X)$:

$$\overline{T}(X) = \{t_i \in \mathcal{T} : T(t_i) \cap X \neq \emptyset\} \tag{2.21}$$

Upper approximations of all documents are calculated using equation (2.21). Method determines k representatives of the whole document set initially. In the next step, method works as follow. For each document $x \in \mathcal{D}$, method computes similarity between representative $R_i, i = 1..k$ and x_i. If similarity is more than a pre-specified parameter, then x_i is assigned to R_i. It again selects representative from each cluster and repeats the process of re-allocation. The method uses cosine and Jaccard similarity measured in the clustering process. Finally, each un-assigned document is allocated to nearest cluster.

De and Krishna [De and Krishna (2004)] proposed rough-approximation based clustering method for web access log database. Novelty of their approach is measuring similarity between two transactions using *Similarity Upper Approximation*, which is the function of upper approximation of tolerance rough set model. Upper approximation of a set of transactions X, $\overline{T}X$ is defined as follows.

$$\overline{T}X = \bigcup_{x \in X} T(x), \text{ where } T(x) = \{y \in \mathcal{D} : sim(x, y) \geq \tau_3\},$$

$sim(.)$ is a similarity function, τ_3 is a user defined parameter. Let $\overline{T}(x)$ be upper approximation of transaction x. Method successively calculates $\overline{T}(x), \overline{T}\,\overline{T}(x)$, and this process continues until two consecutive upper approximations of x are same, which is termed as Similarity Upper Approximation. Similarity Upper Approximation is calculated for each transaction in the dataset. Finally, transactions are merged into a cluster with same similarity upper approximation value. However, method has time complexity of $O(n^2)$.

Tolerance rough set based hierarchical agglomerative clustering method is proposed for sequential dataset [Kumar et al. (2007)]. Upper approximations are utilized to measure similarity between a pair of transactions like previous method. However, this methods accounts number of shared elements between two successive upper approximations of a transaction. Let $\overline{T}(x)$ be the first upper approximation of x. Then, second upper

approximation of x is :

$$\overline{T}\ \overline{T}(x) = \{x_j \in \bigcup_{x_l \in \overline{T}(x)} \overline{T}(x_l) : \text{RelSim}(x, x_j) \geq \sigma\},$$

where $RelSim(x, x_j) = \dfrac{|\overline{T}(x) \cap \overline{T}(x_j)|}{|\overline{T}(x) - \overline{T}(x_j)|}$

Method initially calculates upper approximation of each transaction x_i in the dataset. Let S_i be the upper approximation of x_i. Next, For each transaction x, it calculates next higher order approximations S_i'. If $S_i = S_i'$, then transactions in S_i' are merged into a cluster, otherwise method repeats the process of calculating next higher order approximations and subsequently comparison for possible merging until all transactions form clusters. They showed that their method outperforms classical complete-link clustering method.

S. Kawasaki et al. [Kawasaki et al. (2000)] proposed tolerance rough set model (TRSM) based hierarchical document clustering. The method has two phases. In the first phase, it exploits tolerance rough set model to enrich the document representation in terms of semantics relatedness between documents. Tolerance relation is defined based on the co-occurrence of index terms present in document sets. Weights of each index term in a document is calculated based on the upper approximation of the document. In the second phase, an agglomerative clustering method is applied to the dataset. In each step of agglomeration, it merges two most similar clusters. Here, similarity is calculated between the upper approximations of respective representatives of the clusters instead of average similarity between the documents in both clusters. Representative R_k of a cluster C_k is calculated dynamically. Index terms (features) of R are collected incrementally by analyzing each document in C_k as follow. Initially, $R_k = \emptyset$. For each document $x_j \in C_k$, for each term $t_i \in x_j$, if frequency of the term t_i in C_k is more than a user specified value, then t_i is added to R_k as a term, i.e., $R_k = R_k \cup \{t_i\}$. Then, if it is found that if any document x does not contribute any term to R_k, then maximum weighted term in x is added to R_k. Method demonstrates that TRSM based document representation outperforms VSM (vector-space-model) based representation for the retrieval purpose.

Thesis has a contribution in which tolerance rough set model is utilized.

2.4 Conclusion

This chapter discussed basic background of clustering methods and major clustering methods. Discussion emphasized on the clustering methods which can find arbitrary shaped clusters in large dataset. The chapter also pointed out the research direction of the thesis along with drawbacks of the existing methods. Contribution begins with the next chapter, which presents two distance based hybrid clustering methods for finding clusters of arbitrary shapes and sizes in large dataset.

Chapter 3

A distance based hybrid clustering method for arbitrary shaped clusters in large dataset

3.1 Introduction

Distance based clustering methods usually find clusters of convex shapes. However, classical single-link is a distance based clustering method which can find arbitrary shaped clusters. It is used in many applications such as image segmentation, spatial data mining, geological mapping. The single-link method builds a dendogram where each level represents a clustering of a dataset. A suitable clustering is chosen from the dendogram based on requirements. Selection of a clustering from dendogram can be done by specifying various stopping conditions as follow [Kleinberg (2002)].

i) Distance h stopping condition : Distance between any pair of clusters in the clustering should be greater than h.

ii) k-cluster stopping condition: Number of clusters in the clustering should be k.

iii) Scale α stopping condition : Let ρ^* be the maximum pairwise distance between any two patterns in the dataset. Then, the stopping condition is: distance between any

pair of cluster should be greater than $\alpha\rho^*$, where $0 < \alpha < 1$.

Single-link clustering method with distance-h stopping condition is referred to as *SL* method in the thesis. Krause exploited SL method in computational biology for searching a group of homologous biological sequences [Krause et al. (2005)]. However, SL method has the following drawbacks: (i) time and space complexity of $O(n^2)$, where n is the size of the dataset and (ii) scanning the dataset many times for constructing the distance matrix. Therefore, this method is not suitable for large dataset.

In this chapter, a distance based hybrid clustering method which is suitable for clustering large dataset is proposed. The proposed method is a combination of leaders clustering method (partitional clustering) and SL method. This hybrid method is termed as *leader-single-link* (l-SL) method, (l stands for leaders). In this method, first leaders clustering method is applied on the dataset to derive a set of leaders. Subsequently, SL method is applied on the leaders set to obtain the final clustering. l-SL method produces a flat clustering of the dataset. The contributions of this chapter are as follow.

- Proposed l-SL method can find clusters of arbitrary shapes and sizes.
- The l-SL method scans the dataset once and it is significantly faster than the SL method.
- Theoretically, it is proved that clustering produced by the l-SL method is a refinement [Tremblay and Manohar (1997)] of the clustering produced by SL method.

However, the clustering results produced by l-SL method may deviate from the final clustering produced by SL method. From various experimental studies, it is found that deviations are nominal. To overcome this deviation (if any), a correction scheme is also proposed and termed as al-SL method. The correction scheme has the following features.

- Execution time of the al-SL method is marginally higher than the l-SL method, but significantly faster than the SL method.
- Clustering results produced by the al-SL method and the SL method are identical.

- The *al*-SL method scans the dataset twice. Therefore, it is suitable for large dataset.

The rest of the chapter is organized as follows. Section 3.2 describes a summary of related works. Section 3.3 describes a brief background of the proposed clustering methods. Section 3.4 describes proposed distance based hybrid clustering method (*l*-SL) and a relationship between SL method and *l*-SL method. Section 3.5 describes proposed improvement of the *l*-SL method. Experimental results and conclusions are discussed in Section 3.6 and Section 3.7, respectively.

3.2 Related work

For the sake of readability, a brief summary of distance based clustering methods which can handle large dataset (discussed in Subsection 2.3.3, Subsection 2.3.4 of Chapter 2) is presented here. These distance based clustering methods can be categorized into two classes based on the shapes of clusters (globular or non-globular) produced by them.

- **Globular-shape based clustering:** The Sequential Algorithmic Scheme [Theodoridis and Koutroumbas (2006)] like BSAS, modified BSAS, TTSAS, Max-min algorithm, leaders are highly suitable for clustering large dataset as this scheme scans the dataset twice. However, these methods can find only convexed shaped clusters. Objective of the thesis is to develop clustering methods for arbitrary shaped clusters. Therefore, discussion of other clustering methods which find globular shaped clusters in large dataset [Dash et al. (2003), Zhang et al. (1996)] are not included here.

- **Non globular-shape (Arbitrary-shape) based clustering:** Distance based hierarchical clustering method single-link can find arbitrary shaped clusters. However, it is not suitable for large dataset. Methods are developed to make single-link workable in large dataset. These are discussed briefly.

  Nanni [Nanni (2005)] exploits triangle inequality property of metric space to speed-up single-link clustering methods. Let d be a distance function in a metric space

and x, y and z be three arbitrary points in the space. Then, triangle inequality states that,

$$d(z, x) + d(x, y) \geq d(y, z) \tag{3.1}$$

Rewriting the above equation (3.1), one can obtain the following equation (3.2).

$$|d(x, y) - d(y, z)| \leq d(x, z) \leq d(x, y) + d(y, z) \tag{3.2}$$

In this approach, equation (3.2) is repeatedly used to approximate actual distance between x and z having known the distance between x, y and y, z. The scheme shows a large reduction in distance computations in k-cluster stopping condition of single-link method. The scheme needs to access the dataset many times. Koga et al. [Koga et al. (2007)] proposed a randomized version of single-link method. Here, the scheme utilizes *Locality-Sensitive Hashing* (LSH) [Indyk and Motwani (1998)] for quickly finding nearest cluster of a given cluster. It creates clustering hierarchy of a given dataset. It can find clusters of arbitrary shapes and sizes. However, clustering results are highly influenced by many parameters, which are difficult to estimate in large size real dataset. LSH acceses the datset many times for performing range query. Both of the above approaches keep the whole dataset in the main memory of the computer. Therefore, methods cannot be suitable for large size dataset.

Hybrid clustering approach is a way to speed-up classical clustering methods in large dataset. In this approach, a linear partitional clustering is applied in first phase to obtain a number of representative patterns of the dataset. Next, an agglomerative clustering approach is used to these representatives. Many hybrid clustering methods are found in [Murty and Krishna (1981), Wong (1982), Lin and Chen (2005), Liu et al. (2009)]. These approaches speed up single-link [Murty and Krishna (1981), Wong (1982)] or a variant of single-link [Lin and Chen (2005), Liu et al. (2009)]. However, all these hybrid approaches use k-means as the base clustering method, which scans the dataset many time before it gets converged.

Vijaya et al. [Vijaya et al. (2006)] uses single-scanned leaders method as the base

clustering in a hybrid approach, which is used to reduce the computations of classifier in protein sequence analysis. The hybrid approach produces a hierarchy of given protein sequenceses. However, no theoretical and experimental analysis of clustering results are reported in the paper. Relation between leaders threshold (τ) and the number of final clusters (k) are also not reported.

3.3 Background of the proposed method

Two widely used clustering methods *viz.,* leaders clustering and single-link clustering method are discussed in this section. These two clustering methods have their own advantages and disadvantages. The proposed clustering methods exploit these two clustering methods.

Leaders clustering method

Leaders clustering method is a single data-scan distance based partitional clustering method. For a given threshold distance τ, it produces a set of leaders $\mathcal{L}$ incrementally. For each pattern x, if there is a leader $l \in \mathcal{L}$ such that $||x - l|| \leq \tau$, then x is assigned to a cluster represented by l. The time complexity of leaders clustering is $O(mn)$, where $m = \mid \mathcal{L} \mid$. The space complexity is $O(m)$. However, it can only find convex shaped clusters. It is an order dependent clustering method.

Single-link clustering method

Single-link [Sneath and Sokal (1973), Olson (1995)] is a distance based agglomerative hierarchical clustering method. In single-link, distance between any two clusters is the minimum of distances between all pairs of patterns each of which is from different cluster. It produces a hierarchy of clusterings. In this chapter, minimum inter-cluster distance (h) is considered as the selection criteria to find final clustering from the hierarchy. The single-link method with inter-cluster distance (h) is noted as the SL method and depicted in Algorithm 3.1. The SL method is not scalable with the size of dataset. The time and

space requirements of SL method are $O(n^2)$ and it scans a dataset many times.

Algorithm 3.1 SL($\mathcal{D}$, h)

{$\mathcal{D}$ is a given dataset, h is cut-off distance}
Place each pattern $x \in \mathcal{D}$ in a separate cluster. Let $\pi_1 = \{C_1, C_2, \ldots\ldots, C_n\}$
Compute the inter-cluster distance matrix and set $i = 1$.
while {A pair of clusters $C_x, C_y \in \pi_i$ such that $Distance(C_x, C_y) \leq h$} **do**
  Select two closest clusters C_l and C_m.
  Form a new cluster $C = C_l \cup C_m$.
  Next clustering is $\pi_{i+1} = \pi_i \cup \{C\} \setminus \{C_l, C_m\}$; $i = i + 1$
  Update distances from C to all other clusters in the current clustering π_i.
end while
Output final clustering π_i.

3.4 Proposed clustering method

Proposed *l*-SL clustering method is a hybrid scheme with a combination of above two techniques (i.e. leaders and SL method). The *l*-SL method needs only a single parameter h, (*i.e.* distance between a pair of clusters). Like other clustering methods (k-means, DBSCAN), we assume that the value of the parameter (h) is known before hand. In this scheme, the set of leaders produced by the leaders clustering method is used as a representative set of dataset. Subsequently, the SL method is applied to the representative set to obtain final clustering.

3.4.1 Selection of leaders clustering method and its threshold

There exists a category of clustering method called sequential algorithm. These methods are fast and create a single clustering of a dataset [Theodoridis and Koutroumbas (2006)]. Leaders clustering method [Hartigan (1975), Spath (1980)] is one of this type. Basic Sequential Algorithmic Scheme (BSAS), modified BSAS (MBSAS), Max-min algorithm [Juan and Vidal (2000)], Two-Threshold Sequential Algorithmic Scheme (TTSAS) [Theodoridis and Koutroumbas (2006)] are different variations of the leaders clustering method. However, all these methods either scan dataset more than once (MBSAS, TTSAS) or need more than one parameters (BSAS, TTSAS).

The k-means [MacQueen (1967)] method runs in linear time with the size of dataset. It can be used to select prototypes of a dataset. However, the k-means is applicable to numeric dataset only and scans dataset more than once before convergence.
The leaders clustering method can be applied to any dataset where notion of distance (similarity) is defined. It needs one parameter (τ) and single database scan.

Distance criteria τ for the leader clustering method in l-SL method plays a major role to produce final clustering. It is obvious that τ should be less than that of h. It is observed from experimental results that $\tau \leq h/2$ is required to get clustering results at par with results of SL method applied directly. Minimum execution time of l-SL method is found for $\tau = h/2$. Results obtained using $\tau = h/2$ is explained analytically in subsequent subsections.

3.4.2 The *l*-SL method : A variant of single-link clustering method

The l-SL method works as follows. At first, a set of leaders ($\mathcal{L}$) is obtained applying the leaders clustering method to a dataset using $\tau = h/2$. The leaders set is further clustered using SL method with cut-off distance h which results in clustering of leaders. Finally, each leader is replaced by its followers to produce final clustering. The l-SL method is depicted in Algorithm 3.2.

Algorithm 3.2 l-SL($\mathcal{D}$, h)

{$\mathcal{D}$ is a given dataset, h is cut-off distance}
Apply leaders method with $\tau = \frac{h}{2}$ to $\mathcal{D}$. {Let $\mathcal{L}$ be the output}
Apply SL($\mathcal{L}, h$) as given in Algorithm 3.1. {Outputs a clustering $\pi_{\mathcal{L}}$ of the leaders. }
Each leader in $\pi_{\mathcal{L}}$ is replaced by its followers. {This gives a clustering of $\mathcal{D}$ (say π_l).}
Output π_l.

Time and space complexity of the proposed method are analyzed as follow.

1. Step of obtaining the set of leaders ($\mathcal{L}$) takes time of $O(mn)$, where $m = |\mathcal{L}|$. Space complexity is $O(m)$. It scans dataset once.

2. The SL($\mathcal{L}$, h) has time and space complexity of $O(m^2)$.

Overall running time of l-SL is $O(mn + m^2)$. Since, m is always less than n, the complexity becomes $O(mn)$. Experimentally, we show that l-SL is considerably faster than that of SL method. It is because, SL works with the entire dataset, whereas l-SL works with the leaders set (a subset of the dataset). Space complexity of l-SL method is $O(m^2)$.

3.4.3 Reduction of distance computations in *l*-SL method

The l-SL method calculates distance matrix for leaders in order to apply SL method to the leaders. l-SL method computes $\frac{m * (m-1)}{2}$ distance calculations (assuming that distance function is symmetric). However, these distance computations can be fully avoided if the distance matrix can be formed during generation of leaders method. So, we propose to modify leaders clustering method in such a way that it can output distance matrix along with the set of leaders. Modified Leaders method works as follows.

Distance matrix $D^l[\,][\,]$ and leaders set $\mathcal{L}$ are built incrementally. The very first pattern x_1 of $\mathcal{D}$ is added to $\mathcal{L}$ as the first leader of the dataset. For each pattern $x \in \mathcal{D} \setminus \{x_1\}$, it successively calculates distance between x and leader $l_i \in \mathcal{L}$ and stores the distance to a temporary array $T[\ |\mathcal{L}|\ ]$ until x becomes a follower of an existing leader or all leaders in $\mathcal{L}$ are checked. If x becomes a follower of a leader in $\mathcal{L}$, then temporary array $T[\,]$ is discarded and method repeats the same process with new pattern in the dataset. Otherwise, x becomes a new leader and added it to $\mathcal{L}$. The temporary array $T[\,]$ stores distances from new leader x to all other leaders. $T[\ |\mathcal{L}|\ ]$ is added as a row to distance matrix $D^l[\ |\mathcal{L}|\ ]$. Finally, it outputs distance matrix $D^l[\,][\,]$ and leaders set $\mathcal{L}$. Modified leaders is depicted in Algorithm 3.3.

3.4.4 Relationship between SL and l-SL

Relationship between SL method and l-SL method is formally established in this section. For $x, y \in \mathcal{D}$ and a clustering π, we denote $x \sim_\pi y$ if x and y are in a same cluster of π and $x \not\sim_\pi y$ otherwise.

Algorithm 3.3 Modified Leaders($\mathcal{D}$, τ)

{$\mathcal{D}$: Dataset and τ : leaders threshold distance }

$\mathcal{L} = \{x_1\}$; {x_1 is first pattern in $\mathcal{D}$}
$D^l[\,][\,] \leftarrow 0$; { $D^l[\,][\,]$ is Distance Matrix for leaders }
for each $x \in \mathcal{D} \setminus \{x_1\}$ **do**
  $T[\,|\mathcal{L}|\,] \leftarrow 0$ {$T[\,.]$ is a temporary array}
  for $i = 1$ to $|\mathcal{L}|$ **do**
    Calculate distance between l_i and x. Let it be $d(l_i, x)$
    if $d(l_i, x) \leq \tau$ **then**
      x is added as a follower of l; Break;
    else
      $T[i] = d(l_i, x)$;
    end if
  end for
  if x is not follower of any leader in $\mathcal{L}$ **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
    $D^l[\,|\mathcal{L}|+1\,][\,] \leftarrow T[\,|\mathcal{L}|\,]$
  end if
end for
Output $\mathcal{L}, D^l[\,][\,]$.

Let π and π_l be final clustering produced by SL (distance stopping criteria) and l-SL for same value of h, respectively. Then, $\pi = \pi_l$, if the following two conditions hold.

1. For $x, y \in \mathcal{D}$, if $x \sim_{\pi} y$, then $x \sim_{\pi_l} y$.

2. For $x, y \in \mathcal{D}$, if $x \nsim_{\pi} y$, then $x \nsim_{\pi_l} y$

Following definitions, lemmas and theorem are demonstrated for applicability of above two conditions for π and π_l.

Definition 3.1 (Refinement) *[Tremblay and Manohar (1997)] A refinement of a partition* $\pi_1 = \{B_1^{(1)}, B_2^{(1)}, \ldots, B_p^{(1)}\}$ *of* $\mathcal{D}$ *is a partition* $\pi_2 = \{B_1^{(2)}, B_2^{(2)}, \ldots, B_q^{(2)}\}_{q>p}$ *of* $\mathcal{D}$ *such that for each* $B_j^{(2)} \in \pi_2$ *there is a* $B_i^{(1)} \in \pi_1$ *and* $B_j^{(2)} \subseteq B_i^{(1)}$. □

Lemma 3.1 *If two arbitrary patterns* x *and* y *are in a dataset and SL method satisfies* $x \sim_{\pi} y$ *then l-SL satisfies* $x \nsim_{\pi_l} y$.

Proof: This is proved by contradiction. Any two patterns x and y are not in a cluster according to SL method (i.e., $x \nsim_{\pi} y$) means that there is no sequence of patterns $x, a_1, \ldots, a_p, y$ such that distance between any two successive patterns is less than or equal to h. If this does not hold then x and y must be in a cluster.

Let l_x be a leader such that $x \in \mathit{followers}(l_x)$, similarly let l_y be a leader such that $y \in \mathit{followers}(l_y)$. Suppose there is a sequence of leaders $l_x, l_1, \ldots, l_q, l_y$ such that distance between any two successive leaders in the sequence is less than or equal to h. If so, l_x and l_y are grouped into a cluster according to the l-SL. Since, a leader is also a pattern in a dataset, there is a sequence of patterns $x, l_x, l_1, \ldots, l_q, l_y, y$ such that distance between any two successive patterns is less than or equal to h. This is true for l-SL method, since distance between a pattern and its leader is always less than or equal to $h/2$. So, x and y must be grouped in a cluster according to SL. This is the required contradiction. □

Lemma 3.2 *If two arbitrary patterns x and y are in a dataset and SL method satisfies $x \sim_{\pi} y$, then l-SL method may not satisfy $x \sim_{\pi_l} y$.*

Proof : The following scenario may exist in final clustering. Let x be a follower of a l_x and y be a follower of a l_y. If $||x - y|| \leq h, ||l_x - x|| \leq h/2, ||l_y - y|| \leq h/2$ and $||l_x - l_y|| > h$, clearly, SL method groups these four patterns into a cluster. Since $||l_x - l_y|| > h$, l_x, l_y and their followers are not grouped into a cluster by l-SL (for clarity, this is also shown in Figure 3.1). Hence, $x \nsim_{\pi_l} y$ holds. □

Lemma 3.1 states that if x and y are not grouped into a cluster by SL method then l-SL does not group them into a cluster. Lemma 3.2 states that the number of clusters in the final clustering produced by l-SL method may be more than that of the SL method *i.e.* few clusters produced by l-SL method might be grouped as one cluster in SL method. From the above two Lemmas the following Theorem 3.1 can be derived.

Theorem 3.1 *The clustering output of l-SL method is a refinement of the clustering output of SL method.* □

As per the Theorem 3.1, the cardinality of final clustering generated by l-SL method may be more compared to final clustering generated by SL method for a given dataset, if they

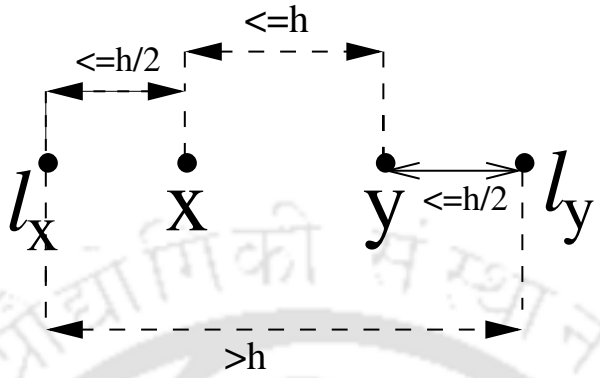


Figure 3.1: A probable representation for Lemma 3.2

consider the same value of h. Experiments with various standard and synthetic dataset show that the deviation (i.e difference in cardinality) is nil or a small value. However, to get exact clustering as that of the SL method, further refinements (i.e. merging of the deviated clusters) of l-SL method is proposed in the following section.

3.5 Augmented *l*-SL (*al*-SL) clustering method

As discussed earlier, a few clusters in the final results of l-SL method may be required to be merged in order to obtain the exact final results as that of the SL method. Let the clustering result of the l-SL be a clustering $\pi_l = \{B_1, B_2, \ldots, B_k\}$. Let (B_i, B_j) be a pair of clusters in π_l. If there exists a pair of patterns $x \in B_i$ and $y \in B_j$ such that $||x - y|| \leq h$, then B_i and B_j must be merged together. In general, one needs to search for all pairs of clusters in π_l to see whether they can be merged. The following Lemma 3.3 shows that it is not required to search all pairs of clusters, except for a few pairs.

Definition 3.2 (Cluster of leaders) *A cluster of leaders $B_i^l = \{l_1, l_2, \ldots, l_k\}$ is a group of k leaders, which are found by the l-SL method.* □

Lemma 3.3 *Let B_i^l and B_j^l be two clusters of leaders as found by the l-SL method which are expanded into clusters of patterns (i.e. each leader is replaced with its followers set) B_i and B_j. If* Distance$(B_i^l, B_j^l) > 2h$, *then* Distance$(B_i, B_j) > h$.

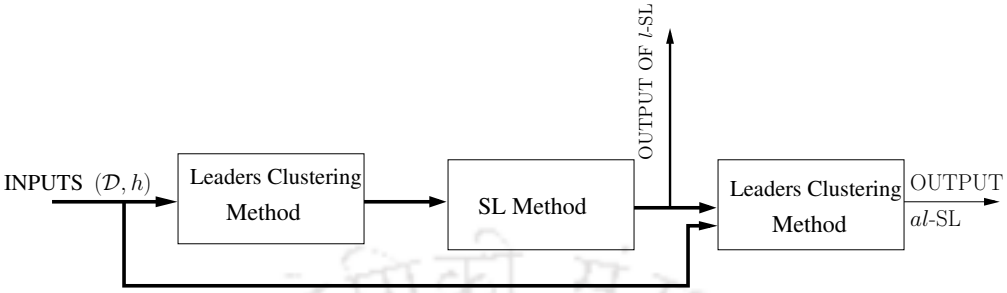


Figure 3.2: Different stages of hybrid clustering methods l-SL and al-SL.

Proof: Let l_i be an arbitrary leader in B_i^l and l_j be an arbitrary leader in B_j^l such that $||l_i - l_j|| > 2h$. Let x be an arbitrary pattern in *followers*(l_i) and y be an arbitrary pattern in *followers*(l_j).

As x is a follower of l_i, we get $||x - l_i|| \leq h/2$. Since $||x - l_i|| \leq h/2$ and $||l_i - l_j|| > 2h$, we get $||x - l_j|| > 1.5h$ (based on triangle inequality).

Similarly, $||l_j - y|| \leq h/2$ and $||x - l_j|| > 1.5h$, we get $||x - y|| > h$. Considering $||x - y|| > h$ satisfies for all x and y, $Distance(B_i, B_j) > h$. □

It is obvious (from the Lemma 3.3) that if $Distance(B_i^l, B_j^l) > 2h$, merging of the clusters B_i and B_j is not possible. On the other hand, as per l-SL method $Distance\ (B_i^l, B_j^l) \leq h$ is not possible (i.e. both of the clusters are in same cluster). So, the clusters B_i^l and B_j^l may be merged, if $h < Distance(B_i^l, B_j^l) \leq 2h$. Therefore, followers of a few leaders in B_i^l and B_j^l need to be searched for possible merging of clusters. It can be mentioned that merging of clusters pair (B_i, B_j) is equivalent to merging of the clusters (B_i^l, B_j^l). Considering all these points, we have augmented l-SL method by incorporating an option of merging of clusters. This method is termed as al-SL method.

Once clustering of leaders $(\pi_{\mathcal{L}})$ is available from l-SL method, al-SL method finds pairs of clusters (B_i^l, B_j^l) such that $Distance(B_i^l, B_j^l) \leq 2h$. Let (l_i, l_j), $l_i \in B_i^l,\ l_j \in B_j^l$ be a pair of leaders such that $||l_i - l_j|| = Distance(B_i^l, B_j^l)$. Next, we identify all leaders $l_x \in B_i^l$ and $l_y \in B_j^l$ which satisfy following conditions, (i) $||l_i - l_y|| \leq 2h$, (ii) $||l_j - l_x|| \leq 2h$.

These leaders l_x, l_y are the potential leaders for possible merging of the clusters (B_i^l, B_j^l).

All such leaders are grouped for further operations. Let L_{B_i} be a set of all l_x leaders and L_{B_j} be a set of all l_y leaders. It implies that followers of the leaders in sets L_{B_i} and L_{B_j} are potential decision maker for merging the cluster-pair B_i^l and B_j^l. The next step is to find all followers of these leaders, this requires the leaders clustering to be executed once more (ordering of patterns in dataset and $\tau = h/2$ to be same as the first time scanning). This time we store followers of all potential leaders for merging.

The last step of possible merging is to find the distance between pair of patterns (i.e. followers) of a pair of clusters (B_i^l, B_j^l) such that $Distance(B_i^l, B_j^l) \leq 2h$. If the distance between two patterns (i.e. followers) of the pair of clusters is less than or equal to h, then the cluster-pair is merged. Different stages of the *l*-SL and *al*-SL methods are summarized in Figure 3.2. The *al*-SL method is given in Algorithm 3.4.

3.5.1 Complexity Analysis

The *al*-SL method has two parts *viz.,* *l*-SL part and the merging part. The complexity analysis of the *l*-SL is discussed in previous section. Here, complexity of the merging part is discussed.

i) Finding potential leaders sets for possible merging of a pair of clusters (B_i^l, B_j^l) can be done in $O(m)$ time. For all pairs of clusters, it takes time of $O(m)$. Space requirement for this step is $O(m^2)$.

ii) Space requirement to store the followers of $\mathcal{S}$ is $O(\sum_{k=1}^{|\mathcal{S}|} |l_k|,\ l_k \in \mathcal{S})$ and the time requirement is $O(mn)$.

iii) Time required for finding minimum distance between a pair of leaders is $O(f^2)$, where f is the average number of followers of a leader. If m_a is the average number of potential leaders responsible for merging a pair of clusters, then the time required to find minimum distances for all pair of clusters is $O(m_a^2 f^2)$.

Time complexity of the merging part is $O(m)+O(mn)+O(m_a^2 f^2)$, *i.e.* $O(mn+m_a^2 f^2)$ as $m_a < m, m_a^2 f^2 \approx mn$. So, time complexity of this part is approximately

Algorithm 3.4 *al*-SL($\mathcal{D}$, h) /*$\mathcal{D}$ is a given dataset, h is cut-off distance */

Apply leaders method with $\tau = h/2$ to $\mathcal{D}$. {Outputs $\mathcal{L}$, set of leaders.}
Apply SL($\mathcal{L}, h$) as given in Algorithm 3.2. {Outputs $\pi_{\mathcal{L}}$, a clustering of $\mathcal{L}$.}
$\mathcal{S} = \emptyset$ {*The merging part begins*}
for each pair of clusters (B_i^l, B_j^l) in $\pi_{\mathcal{L}}$ with *Distance*$(B_i^l, B_j^l) \leq 2h$ **do**
  Identify a pair of leaders (l_i, l_j) such that $||l_i - l_j|| = Distance(B_i^l, B_j^l))$
  $L_{B_i} = \emptyset$; $L_{B_j} = \emptyset$
  for each $l_x \in B_i^l$ **do**
    if $||l_x - l_j|| \leq 2h$ **then**
      $L_{B_i} = L_{B_i} \cup \{l_x\}$
    end if
  end for
  for each $l_y \in B_j^l$ **do**
    if $||l_y - l_i|| \leq 2h$ **then**
      $L_{B_j} = L_{B_j} \cup \{l_y\}$
    end if
  end for
  $\mathcal{S} = \mathcal{S} \cup L_{B_i} \cup L_{B_j}$. { $\mathcal{S}$, set of potential leaders for possible merging}
end for
if $\mathcal{S} \neq \emptyset$ **then**
  Apply leaders method with $\tau = h/2$ and store followers of leaders in $\mathcal{S}$. {$\mathcal{D}$ *is scanned for second time in same order*}
  for each pair of clusters (B_i^l, B_j^l) in $\pi_{\mathcal{L}}$ such that *Distance*$(B_i^l, B_j^l) \leq 2h$ **do**
    for each pair of potential leaders (l_a, l_b) such that $l_a \in L_{B_i}$ and $l_b \in L_{B_j}$ **do**
      Find two nearest followers (x, y) ; $x \in l_a$ and $y \in l_b$
      if $||x - y|| \leq h$ **then**
        Merge B_i^l and B_j^l into a single cluster; break;
      end if
    end for
  end for
end if {*The merging part ends*}
$\pi_{\mathcal{L}}$ is converted into π_{alSL} by replacing each leader by its followers. Output π_{alSL}.

$O(mn)$. Space complexity is $O(m^2) + O(|\mathcal{S}| * f) \approx O(m^2)$ (since, $|\mathcal{S}|$ is very less than m). Overall time complexity of *al*-SL method becomes $O(mn)$ and overall space complexity is $O(m^2)$.

As discussed in section 3.3, the results of leaders clustering depend on the scanning order of the dataset. The same is true for *l*-SL method. However, clustering results of *al*-SL method are independent of the scanning order. It is proved in the following subsection.

3.5.2 Relationship between SL and al-SL

In this section, we formally establish that clustering result produced by the *al*-SL method is same as that of the SL method and result of the *al*-SL is independent of the scanning order of the dataset.

Definition 3.3 (Reachable) *A pattern $x \in \mathcal{D}$ is reachable from another pattern $y \in \mathcal{D}$, if there exists a sequence of patterns $x_1, x_2, \ldots, x_p$, where $x_1 = y, x_p = x$ such that $||x_i - x_{i+1}||_{i=1..p-1} \leq h, \quad x_i \in \mathcal{D}$.* □

This *reachable* relation $R \in \mathcal{D} \times \mathcal{D}$ is an equivalence relation.

Lemma 3.4 *The clustering results produced by the al-SL method is same as that of the SL method.*

Proof : Let $\pi_{SL} = \{C_1, C_2, \ldots, C_k\}$, where $C_i \subseteq \mathcal{D}$, be the clustering result produced by SL method. $C_i = \{x_i \in \mathcal{D} : x_i \quad \text{is reachable from} \quad x_j \in C_i\}$ is a cluster of $\mathcal{D}$. The π_{SL} is a partition of $\mathcal{D}$. Therefore, there exists a unique equivalent relation R_1 on $\mathcal{D}$ and each $C_i \in \pi_{SL}$ is an equivalence class of $\mathcal{D}$ by the relation R_1, $\quad C_i = [x_i]_{R_1} = \{x_j \in \mathcal{D} \mid x_i R_1 x_j\}$. Clearly, R_1 is a reachable relation of $\mathcal{D}$.

The *al*-SL method initially creates a partition $\pi_{\mathcal{L}} = \{B_1^l, B_2^l, \ldots, B_p^l\}$ of leaders set $\mathcal{L}$, where B_i^l is a cluster of leaders. $\pi_l = \{B_1, B_2, \ldots, B_p\}$ can be a partition of the dataset if each leader $l \in B_i^l \in \pi_{\mathcal{L}}$ is replaced by its followers (Note that each B_i^l converted into $B_i \in \pi_l$ with a mapping $f : \pi_{\mathcal{L}} \rightarrow \pi_l$ such that $f(B_i^l) = B_i$). However, $\pi_l \neq \pi_{SL}$ due to Theorem 3.1. There may exist a pair of patterns $x \in B_i, y \in B_j (i \neq j)$ such that $||x - y|| \leq h$ (Lemma 3.2). Again, according to Lemma 3.3, there cannot be any pair of patterns $x \in B_i, y \in B_j$ such that $||x - y|| \leq h$, if $Distance(B_i^l, B_j^l) > 2h$. Therefore, *al*-SL exhaustively searches followers of all potential leaders of each pair (B_i^l, B_j^l) with $Distance(B_i^l, B_j^l) \leq 2h$ (Algorithm 3.4). If there exists a pair of followers (patterns) x, y of any potential leaders pair of (B_i^l, B_j^l) such that $||x - y|| \leq h$, then *al*-SL method merges (B_i^l, B_j^l). Finally, it produces a clustering $\pi_{alSL} = \{B_1, B_2, \ldots, B_k\}_{k<=p}$. The π_{alSL} is also a partition of $\mathcal{D}$ and corresponds an equivalence relation R_2. Therefore, $B_i \in \pi_{alSL}$ is an

equivalent class of $\mathcal{D}$ by R_2. More formally, $[x_i]_{R_2} = B_i = \{x_j \in \mathcal{D} : x_j$ is reachable from $x_i \in B_i\}$. Clearly, R_2 is also a reachable relation of $\mathcal{D}$. Therefore, $R_2 = R_1$. It follows that $\pi_{SL} = \pi_{alSL}$. □

Theorem 3.2 *Clustering results produced by the al-SL method is independent of the scanning order of the dataset by leaders clustering method.*

Proof : For the sake of simplicity, we consider two different scanning orders of the dataset $\mathcal{D}$ by leaders clustering method. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_{m1}\}$ and $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_{m2}\}$ be two sets of leaders obtained in two different scanning orders. Having applied the *al*-SL method to $\mathcal{L}$ and $\mathcal{L}'$ separately, we obtain two partitions $\pi_{alSL} = \{B_1, B_2, \ldots, B_k\}$ and $\pi'_{alSL} = \{B'_1, B'_2, \ldots, B'_{k'}\}$, respectively. We have to show that $\pi_{alSL} = \pi'_{alSL}$.
From Lemma 3.4, we can say that π_{alSL} and π'_{alSL} both correspond to same equivalence relation, which is a reachable relation on $\mathcal{D}$. Therefore, $\pi_{alSL} = \pi'_{alSL}$. □

3.5.3 Estimating the value of h

Proposed clustering methods *l*-SL and *al*-SL need a parameter h. Two approaches for finding the value of h are reported as follow.

- **First approach:** Many *domain experts* know approximate distances between natural clusters. For example, cell biologists might have an idea of average distance between the chromosomes. Medical practitioner might have some knowledge about the average distances between bones in different parts of the body in x-ray image.

- **Second Approach:** This approach selects $O(\sqrt{n})$ patterns from a dataset randomly and single-link clustering method is applied to these selected patterns. From dendogram generated by the single-link method, one can find *life time* for each clustering. *Life time (LT)* [Jain and Martin (2005)] of a clustering π_i is defined as follow.

$$LT(\pi_i) = (d_i - d_{i+1}),$$

where d_i, d_{i+1} are the distances between two closest clusters in π_i and π_{i+1} clustering respectively;

π_i, π_{i+1} are clusterings obtained from two consecutive layers of the dendogram. *Maximum life time* clustering is that for which life time is maximum.

This *maximum life time* can be considered as the value of h.

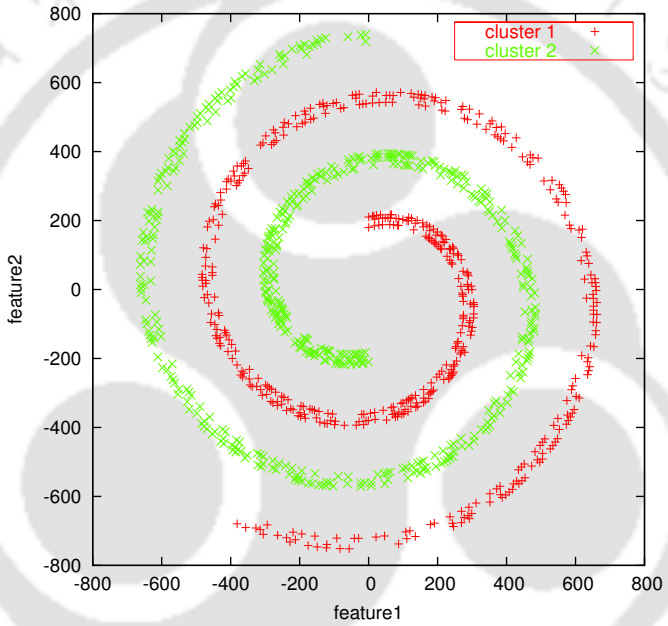


Figure 3.3: Spiral dataset.

Table 3.1: Dataset used in experiments

Dataset	# Patterns	# Features
DNA	2000	180
Banana	4900	2
Spiral (Synthetic)	3330	2
Pendigits	7494	16
Shuttle	58000	9
GDS10	23709	28
Circle4 (Synthetic)	28000	2

3.6 Experimental Evaluation

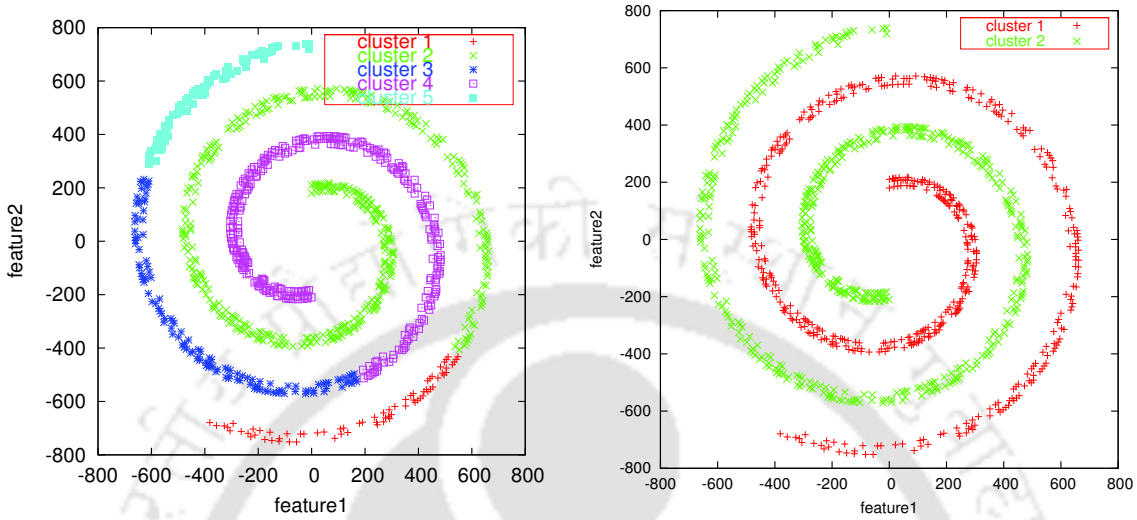
In this section, the proposed methods (*l*-SL and *al*-SL) are evaluated experimentally. The method proposed in [Nanni (2005)] keeps dataset as well as distance-matrix for the entire dataset in primary memory. The scheme proposed in [Koga et al. (2007)] needs many parameters to be adjusted to produce same clustering as that of the single-link method, which are difficult to determine. This method also requires whole dataset to be kept in primary memory. Therefore, these methods are not suitable for large dataset. The hybrid clustering method in [Vijaya et al. (2006)] is used to reduce computational burden of classification. Clustering results are not analyzed experimentally. There is no relation between the leaders threshold (τ) and the number of clusters (k). Therefore, comparisons with the proposed schemes are not reported here.

So, clustering results of the *l*-SL, *al*-SL methods with that of the SL method are compared with help of Rand Index (RI) [Rand (1971)]. Rand Index (RI) is defined as follows. Given a dataset $\mathcal{D}$ of n patterns and let π_1 and π_2 be two clusterings of $\mathcal{D}$. RI is a similarity measure between a pair of clusterings, *i.e.*

$$RI(\pi_1, \pi_2) = \frac{a_1 + e_1}{a_1 + b_1 + c_1 + e_1}$$

where, a_1 is the number of pairs of patterns belonging to a cluster in π_1 and to a same cluster in π_2, b_1 is the number of pairs belonging to a same cluster in π_1 but to different clusters in π_2, c_1 is the number of pairs belonging to different clusters in π_1 but to a same cluster in π_2, and e_1 is the number of pairs of patterns belonging to different clusters in π_1 and to different clusters in π_2.

Rand Index (RI) are computed between the clusterings produced by *l*-SL and SL method, and between the clusterings produced by *al*-SL and SL method for various values of h. To prove effectiveness of the proposed methods, experiments are conducted with standard as well as large dataset. Two synthetic and five real world dataset



(a) Clusters produced by *l*-SL method (b) Clusters produced by *al*-SL Method

Figure 3.4: Experiments with Spiral dataset

(`http://archive.ics.uci.edu/ml/`, `http://www.ncbi.nlm.nih.gov/geo/`) are used after eliminating class labels. A brief description of the dataset is given in Table 3.1. Plot of a synthetic dataset (Spiral Data) is shown in Fig. 3.3.

3.6.1 Experiments with standard dataset

All methods (*l*-SL, *al*-SL and SL) are implemented using C language and executed on Intel Core 2 Duo CPU (3.0 GHz) with 2 GB RAM IBM PC. SL method is implemented using Nearest neighbor array [Olson (1995)].

The *l*-SL and the *al*-SL methods are tested with Spiral dataset and results are reported in Table 3.2. The *l*-SL method produces five (5) clusters of arbitrary shapes (Fig. 3.4(a)). However, it cannot find exact number of clusters (*i.e.* 2) of the dataset. On the other hand, the *al*-SL method finds exactly two arbitrary shaped clusters of the dataset (Fig. 3.4(b)). Further, the *l*-SL method is more than 160 times faster than that of the SL method. Clustering results of the *l*-SL method ($RI = 0.808$) is not same as that of the SL method ($RI = 1.000$) (Table 3.2). The *al*-SL method produces same clustering results as produced by the SL method ($RI = 1.000$). However, execution time of the *al*-SL method is slightly

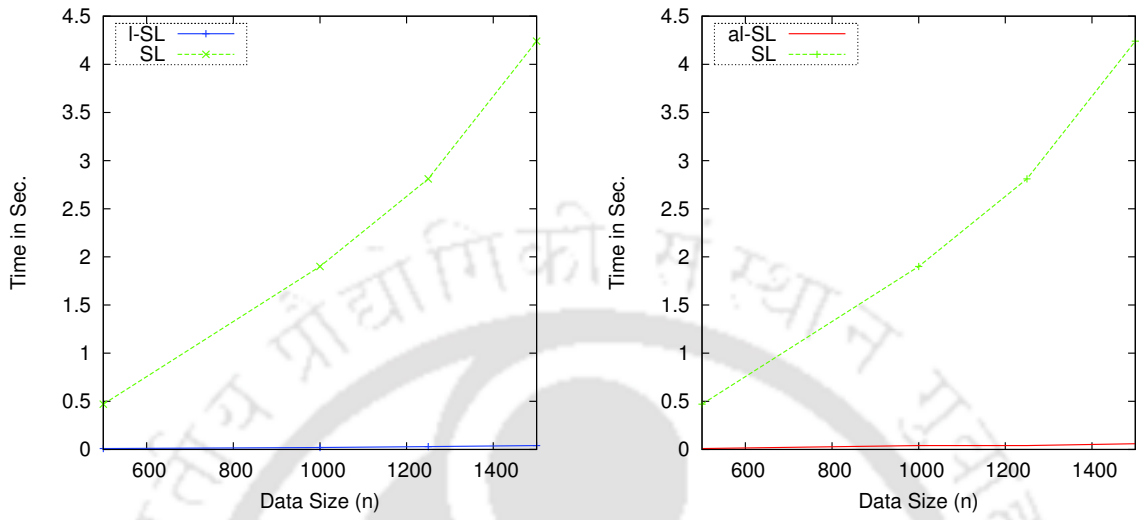
Table 3.2: Experimental Results of hybrid methods with standard dataset

Dataset	Distance(h)	Method	Time (in Sec.)	Rand Index (RI)
Spiral (Synthetic)	50.0	l-SL	0.08	0.808
	50.0	al-SL	0.12	1.000
	50.0	SL	20.85	–
	40.0	l-SL	0.12	1.000
	40.0	al-SL	0.14	1.000
	40.0	SL	19.32	–
DNA	11.00	l-SL	6.68	1.000
	11.00	al-SL	9.61	1.000
	11.00	SL	250.28	–
	9.00	l-SL	6.72	1.000
	9.00	al-SL	9.61	1.000
	9.00	SL	232.84	–
Banana	0.70	l-SL	0.01	0.999
	0.70	al-SL	0.04	1.000
	0.70	SL	43.55	–
	0.50	l-SL	0.02	0.998
	0.50	al-SL	0.05	1.000
	0.50	SL	42.60	—
	0.30	l-SL	0.07	0.997
	0.30	al-SL	0.11	1.000
	0.30	SL	41.68	–

higher than the l-SL method but more than 100 times faster than that of the SL method with $h = 50$ and $h = 40$ (Table 3.2).

To show performance of the proposed methods with variable dataset sizes, experiments are performed using Spiral dataset. In Fig. 3.5(a), Fig. 3.5(b), curves for l-SL and al-SL methods are almost parallel and close to horizontal axis. Both plots show that l-SL and al-SL methods grow linearly with the size of the dataset, whereas SL method grows quadratically for a value of $h = 50$. Time taken by three methods (SL, l-SL and al-SL) are shown in Figure 3.6. It can be concluded that proposed schemes are more scalable and effective for large dataset.

To show the time requirement of the merging process in al-SL method, experiments are also performed with the Spiral dataset. In the merging part of the al-SL method, potential leaders are searched for possible merging of clusters. Figure 3.7 shows the percentage of



(a) Execution time of *l*-SL and SL methods (b) Execution time of *al*-SL and SL methods

Figure 3.5: Time taken by *l*-SL, *al*-SL and SL methods for Spiral Dataset with $h = 50$.

total leaders to be searched with respect to distance (h). It shows that *al*-SL method selects only few number of leaders ($< 10\%$) for possible merging in wide range of h values.

For the **DNA dataset**, *l*-SL and *al*-SL both produce same clustering results ($RI = 1.000$) as produced by the SL method. Both of these methods are significantly faster than that of the SL method.

In case of the **Banana dataset**, *l*-SL could not produce same results as produced by the SL method, but *l*-SL is significantly faster than that of the SL method. The *al*-SL method produces the same results as produced by the SL method. The *al*-SL method takes slightly more time compared to the *l*-SL method.

3.6.2 Experiments with large dataset

Experiments are also conducted with large dataset on Intel Xeon Processor ($3.6GHz$) with 8 GB RAM IBM Server to show the effectiveness of the proposed clustering methods in large dataset. Experimental results are reported in Table 3.3.

Synthetic dataset **Circle4** contains four (4) circles of same radius and they are well

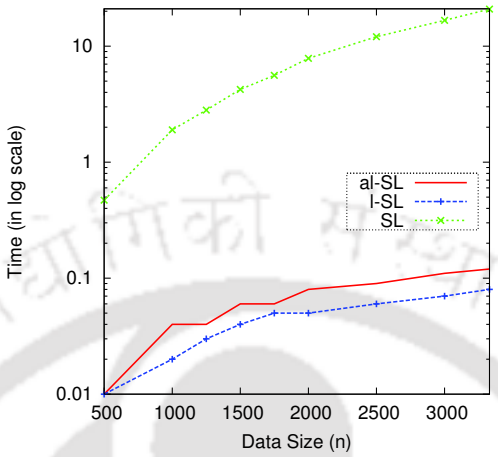


Figure 3.6: Execution time of the *l*-SL, *al*-SL and SL methods for Spiral Dataset at h=50.

separated from each other. The results of the *l*-SL method are very close to the result of the SL method. *l*-SL method is significantly faster than SL method. *al*-SL method produces same clustering as that of SL method. However, its execution time is marginally higher than *l*-SL method. Similar results are also observed for other three real-world dataset (**Pendigits, Shuttle and GDS10**).

Pendigits is a handwritten digits dataset [Frank and Asuncion (2010)]. In this dataset, patterns are distributed over ten classes $(0, 1, \ldots, 9)$. Training set (eliminating class labels) is used to show the effectiveness of the proposed methods with real-world dataset. Clustering results produced by *l*-SL method is close ($RI = 0.999$, with $h = 90$, $RI = 0.993$ with $h = 70$) to the clusterings produced by the SL method (Table 3.3). The *al*-SL method produces identical (same) clustering results ($RI = 1.000$) as produced by the SL method (Table 3.3). However, *l*-SL and *al*-SL methods are more than *two* orders of magnitude faster than the SL method.

Shuttle Dataset: This dataset has 9 integer valued attributes of $58,000$ patterns distributed over 7 classes (after merging training and test sets). Class labels are eliminated

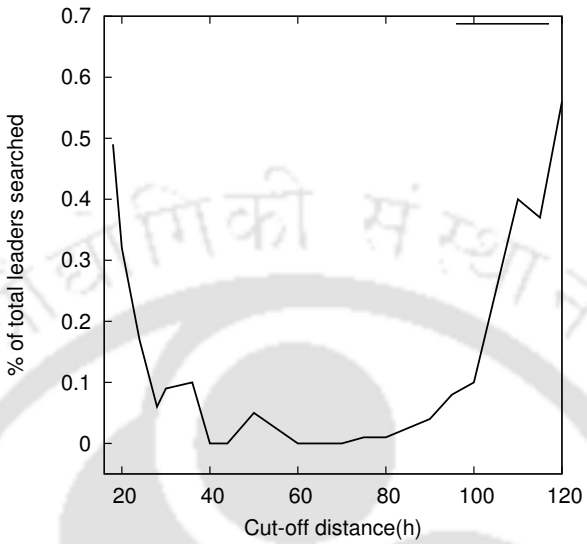


Figure 3.7: The percentages of leaders searched by the *al*-SL method for Spiral Data Set.

from the all patterns. Figure 3.8 shows execution time of the three methods for variable dataset size. Dataset size varies from $5,000$ to $40,000$ for $h = 0.01$ with Shuttle dataset. When dataset size is more than $40,000$, execution could not be completed due to memory shortage. To store distance matrix for the entire dataset, SL method consumes $58000 \times 58000 \times 4 = 12.54GB$ memory. Therefore, experimental results of SL method with whole dataset are not reported. Whereas, *l*-Sl and *al*-SL method produce results within 20 seconds for the whole dataset. Rand Index (RI) between the results of *l*-SL and the results of *al*-SL methods for this dataset are reported in Table 3.3.

From GEO database (`http://www.ncbi.nlm.nih.gov/geo/`), we select a dataset named **GDS10**. Patterns with missing values are removed from this dataset. Finally, we use $23,709$ out of $39,114$ patterns for the experiments. The *l*-SL and *SL* methods are more than *three* and *four* orders of magnitude faster than the SL method with $h = 700$.

For all dataset, similar trends are found (as discussed and proved theoretically) i.e. *l*-SL is faster with approximate SL clustering results. *al*-SL is slightly slower than *l*-SL method

Table 3.3: Experimental Results with large dataset

Data Set	Distance(h)	Method	Time(in Sec.)	Rand Index (RI)
Pendigits	90.0	l-SL	0.46	0.999
	90.0	al-SL	0.79	1.000
	90.0	SL	392.59	–
	70.0	l-SL	1.38	0.993
	70.0	al-SL	2.14	1.000
	70.0	SL	430.46	–
Shuttle	0.02	l-SL	9.13	0.999
	0.02	al-SL	19.38	———
	0.02	SL(40,000)	6929.77	
	0.01	l-SL	9.32	0.999
	0.01	al-SL	20.27	———
	0.01	SL(40,000)	6929.77	
GDS10	700	l-SL	0.50	0.999
	700	al-SL	1.15	1.000
	700	SL	4105.35	-
	900	l-SL	0.26	0.999
	900	al-SL	0.62	1.000
	900	SL	4105.35	-
Circle4 (Synthetic)	0.08	l-SL	19.65	0.995
	0.08	al-SL	36.02	1.000
	0.08	SL	948.26	–
	0.20	l-SL	19.35	1.000
	0.20	al-SL	20.6	1.000
	0.20	SL	960.25	–

but significantly faster than SL method with exact SL clustering results.

Considering the experimental and theoretical results, it is clear that the proposed methods are significantly fast and suitable for large dataset.

3.7 Conclusions

The SL method can find arbitrary (non-convex) shaped clusters in a dataset. However, SL method scans dataset many times. It has time and space complexity of $O(n^2)$. So, SL method is not suitable for large dataset. In this chapter, we proposed two clustering methods to detect arbitrary shaped clusters for large dataset. The first method, l-SL

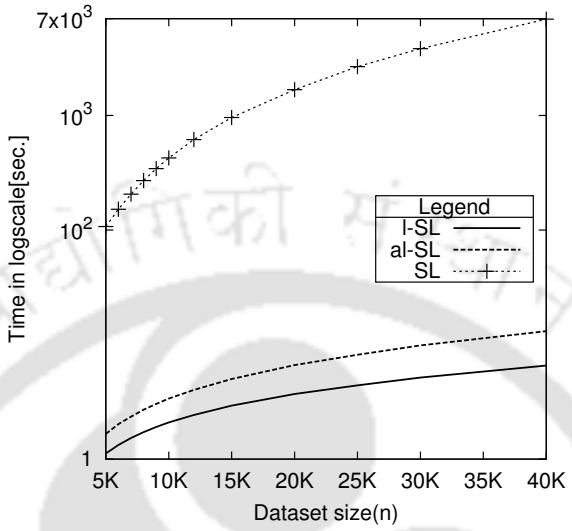


Figure 3.8: Execution time of the *l*-SL, *al*-SL and SL method for Shuttle Dataset.

takes considerably less time compared to that of the SL method and scans the dataset only once. Clustering results produced by *l*-SL is very close to that of the SL method. So, *l*-SL is suitable for the applications where faster and approximate clustering suffice.

The second method, *al*-SL produces exact clustering results as produced by the SL method. Execution time of the *al*-SL method is slightly more than that of the *l*-SL method. Experimental results confirm that the proposed methods are fast and suitable for large dataset.

The *al*-SL method can not produce hierarchical structure of a dataset. Therefore, next chapter proposes a data summarization technique for speeding up hierarchical clustering method in large dataset.

Chapter 4

A new data summarization for hierarchical single-link method in large dataset

4.1 Introduction

One important category of clustering approaches is hierarchical clustering. Hierarchical methods output hierarchical structures of a given dataset. However, these methods assume that entire dataset remains in main memory of the machine during processing time. These methods scan dataset multiple times to produce final clustering results. These are potentially severe problems for cluster analysis in large dataset. One remedy for those problems is to create a summary of a given dataset and the summary is subsequently used to speed up classical clustering methods in large dataset [Zhang et al. (1996), Bradley et al. (1998), DuMouchel et al. (1999), Breunig et al. (2001)].

The Single-link, OPTICS are well known hierarchical methods, which can find arbitrary shaped clusters in dataset. Density based OPTICS [Ankerst et al. (1999)] clustering method outputs an *ordering* of datapoints called *reachability plot*, which represents density based hierarchical structures of a given dataset. It needs two important input

parameters: ϵ ($\in \mathbb{R}$), $Minpts$ ($\in \mathbb{N}$) as that of the density based DBSCAN method. OPTICS clustering can be seen as multiple runs of DBSCAN method with different input parameters $\epsilon' <= \epsilon$. It scans a dataset multiple times for finding range queries of all patterns in the dataset. We restrict our discussion to only one density based method as focus of this chapter is to speed up a distance based hierarchical clustering method, which can find arbitrary shaped clusters in large dataset.

Distance based single-link [Sneath and Sokal (1973)] clustering method outputs a dendogram, which represents hierarchical clustering structures of a given dataset. The single-link method does not need any input parameter to produce dendogram. However, it needs to scan a dataset many times for computing inter-patterns distances of the data. Therefore, both the methods are not suitable for large dataset.

In this chapter, a new summarization scheme termed *data sphere (ds)* is proposed to speed up single-link hierarchical clustering method in large dataset. The *data sphere* utilizes leaders clustering method to collect sufficient statistics of a given large dataset. Single-link clustering method is modified to work with *data sphere*. Modified clustering method is termed as summarized single-link (SSL). The SSL method is considerably faster than the single-link method applied directly to dataset and clustering results produced by SSL method is close to the clustering results produced by single-link method. The SSL method outperforms single-link using data bubble [Breunig et al. (2001)] as a summarization scheme at both in clustering results and computation time. The SSL method also outperforms single-link using data bubble [Zhou and Sander (2003)] at computation time. To speed up proposed summarization scheme, a technique is introduced to reduce a large number of distance computations in leaders clustering method.

Following contributions are made in this chapter.

- New data summarization scheme termed *data sphere* is proposed for distance based hierarchical method to work with large dataset.
- A technique is introduced to reduce number of distance computations in leaders clustering method. Triangle inequality property of metric space is utilized in the reduction process. Therefore, the technique speeds up *data sphere* and SSL method.

- A distance measure is introduced which is suitable for finding arbitrary shaped clusters in dataset.

- Clustering results produced by SSL method on synthetic and real dataset are analyzed using *Rand Index and Purity.*

The rest of the chapter is organized as follows. Section 4.2 describes a brief summary of related research works. Section 4.3 describes proposed summarization scheme and proposed clustering method (termed as SSL method). Experimental evaluations and conclusion are discussed in Section 4.4 and Section 4.5, respectively.

4.2 Related work

A brief summary of different data summarization schemes are presented here (discussed in Subsection 2.3.6 of Chapter 2). An elaborate discussion is also reported on the data summarization schemes closely related to the scheme proposed in this chapter.

As discussed in Chapter 2, following data summarization schemes are popularly used for clustering large dataset.

- **Clustering Feature [Zhang et al. (1996)]:** One of the widely used data summarization/compression schemes is *Clustering Feature (CF)*, which was proposed to speed up average-link clustering method in large dataset [Zhang et al. (1996)]. The *CF* utilizes vector space (Euclidean space) properties to store summarized information of k data points $\{\overrightarrow{X_i}\}_{i=1..k}$. A *CF* is defined as $CF = (k, \overrightarrow{LS}, ss)$, where $\overrightarrow{LS} = \sum_{i=1}^{k} \overrightarrow{X_i}, ss = \sum_{i=1}^{k} \overrightarrow{X_i}^2$. CF values of sub-clusters are organized incrementally in a height-balanced tree called *CF tree*. Leaf nodes of the CF tree represent summary of the whole dataset. However, it is suitable only for k-means types of clustering methods. Because, a centroid is the representative of a cluster in k- means and the centroid can easily be calculated from CF-values. Bradley [Bradley et al. (1998)] also proposed a summarization scheme utilizing Clustering Features to speed up k-means clustering method. Distance between a pair of centroid does not properly

reflect the effective distance between patterns of two groups. Therefore, Clustering Features cannot be used directly for finding arbitrary shaped clusters.

- **Data squashing [DuMouchel et al. (1999)]:** Data squashing initially partitions a given dataset into a number of bins. Bins can be constructed in original space or transformed space. Different order moments like means, minima, maxima, second order moments, third order moments, fourth order moments are calculated for a group of datapoints in each bin. A representative is determined for points in a bin using the computed moments in the next stage. Finally, clustering methods are applied to the representative points. Data squashing outperforms random sample approach. However, scheme is useful for k-means types of clustering methods. Number of bins grows exponentially with dimensions.

- **Data Bubble [Breunig et al. (2000), Breunig et al. (2001)]:** A compressed representation of a subset $X = \{X_i\}_{i=1..k}$ of a dataset is called *Data Bubble*. A Data Bubble utilizes Clustering Features efficiently to collect summarized information of $X = \{X_i\}$. Let $CF = (k, \overrightarrow{LS}, ss)$ be the Clustering Features of X. Then, Data Bubble corresponding to X can be expressed as $B = (k, \overrightarrow{M}, e)$, where M is the representative pattern of X and e is the extent (spread) of X (Figure 4.1). The parameters M and e are calculated using CF values as follow.

$$\overrightarrow{M} = \frac{\overrightarrow{LS}}{k}; \quad e = \sqrt{\frac{2 * k * ss - 2 * \overrightarrow{LS}^2}{k * (k-1)}}$$

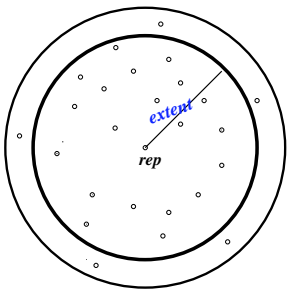


Figure 4.1: A Data Bubble

This summarization scheme works in two stages. In the first stage, a number of seed patterns are selected randomly. Each seed pattern collects CF-triplet incrementally in the next stage. Each pattern of the dataset updates CF of nearest seed pattern. Finally, each seed pattern with its CF is represented by a Data Bubble $B = (k, \overrightarrow{M}, e)$. Set of Data Bubbles is the summary of the given dataset. Density based hierarchical clustering method OPTICS is modified to work with Data Bubbles in large dataset. OPTICS always explores closest distance data point from already explored points. However, distance between centers of a pair of Data Bubbles does not reflect the distance between points of the Data Bubbles. Therefore, a suitable measure is introduced in [Breunig et al. (2000)] to calculate effective distance between a pair of Data Bubbles.

Markus et al. [Breunig et al. (2001)] pointed out that clustering structure detoriates significantly with the increase of compression ratio (size of a dataset to the number of representative objects of the dataset) if OPTICS method is applied to the Data Bubbles using distance measures given in [Breunig et al. (2000)]. Distance between a pair of Data Bubbles and definition of Data Bubble are modified in [Breunig et al. (2001)]. Distance between a pair of Data Bubbles is approximately the distance between two closest points each of which is taken from different Data Bubble.

- **Data Bubble [Zhou and Sander (2003)]:** Data Bubble approaches proposed in [Breunig et al. (2000), Breunig et al. (2001)] speed OPTICS method up significantly in large dataset. However, both approaches suffer from the following drawbacks:

  - CF is the basis of both approaches. Therefore, approaches cannot be used in non vector (categorical) data.

  - The approaches may underestimate effective distance between a pair of bubbles in some cases. Let r_1 and r_2 be two seed patterns of Data Bubbles O_1 and O_2, respectively. Patterns r_1 and r_2 happen to be close to each other (neighbor) and let distance between them be t. Each point x between r_1 and r_2 in the

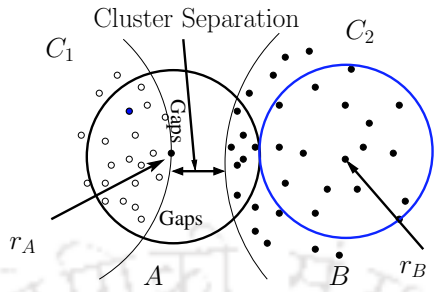


Figure 4.2: Data bubble A has a "gap" in the direction of data bubble B

feature space ($||x - r_1/r_2|| <= t$) will either be distributed to O_1 or O_2 based on the close proximity of x from r_1 or r_2. It may happen that O_2 belongs to a cluster C_2 and few patterns of C_2 ($\{x \mid ||x - r_1|| < t/2, ||x - r_2|| < t\}$) are assigned to O_1. Again, if O_1 contains patterns (x such that $||x - r_2|| > t$) from cluster C_1 also, then cluster separation ("gap") between clusters C_1 and C_2 becomes invisible to data bubble O_1. Therefore, distance between two data bubbles becomes close to zero and O_1 seems to be part of the the cluster C_2 and finally C_1 gets merged with cluster C_2 (Figure 4.2)

In order to overcome both the drawbacks of the Data Bubbles, Zhou and Sander [Zhou and Sander (2003)] introduced "directional" notion to the data bubble. Basic idea of the directional data bubble is to find *extents* (spread) of each Data Bubble in the direction of other Data Bubbles. Let r_1 and r_2 be two seed patterns of Data Bubbles O_1 and O_2, respectively. The extent of O_1 in the direction of O_2 is measured based on the relative positions of patterns in O_1 with respect to O_2. More formally, *extent* of O_1 w.r.t. O_2 is given in equation (4.1) as follows.

$$extent_{O_1}^{(O_2)} = \min\{||r_1 - r_2|| - \min_{x \in O_1}\{||r_2 - x||\}, \quad avg + 2 * stdv\}, \tag{4.1}$$

where, avg and $stdv$ are the minimum of the average and standard deviation of the distances of objects in O_1 in the direction and the reverse direction of O_2. Average (d_{avg}) and standard deviation (d_σ) of distances can easily be computed from the information collected incrementally. A Data bubble O_1 collects information in the

form (k, lsD, ssD), where k is the number of objects in the directions of another data bubble O_2, $lsD = \sum_{1..k} d_i$, $ssD = \sum_{1..k} d_i^2$, $d_i = ||r_1 - x||$, x is a pattern in O_1 in the direction of O_2. Then, average (d_{avg}) and standard deviation (d_σ) of distances are calculated using equation (4.2).

$$d_{avg} = \frac{lsD}{k}; \quad d_\sigma = \sqrt{\frac{k * ssD - (lsD)^2}{k^2}} \tag{4.2}$$

Distance between a pair of Data Bubbles O_1 and O_2 is determined by the positions of representatives and *extents* of the Bubbles (equation (4.3)).

$$dist(O_1, O_2) = ||r_1 - r_2|| - extent(O_1)^{(O_2)} - extent(O_2)^{(O_1)} \tag{4.3}$$

Directional data bubble produces better clustering results compared to the other approaches of the data bubbles in expense of little computational cost.

4.2.1 Data Bubbles and Distance based Clustering Method

As discussed earlier in this Section, data bubble approaches [Breunig et al. (2000), Breunig et al. (2001), Zhou and Sander (2003)] are used to speed up OPTICS clustering method in large dataset. However, it is not studied whether data bubble can be suitable for speeding up distance based single-link method in large dataset.

Data bubbles found in [Breunig et al. (2000), Breunig et al. (2001)] may not be suitable for speeding up distance based single-link method. This is due to the following artifacts.

- In single-link clustering method, distance between two groups (clusters) of data points is the distance between two closest points, each of which is from different clusters. However, effective distance between a pair of well separated data bubbles is computed using the *extent*s and representative points of them. These Summarization schemes [Breunig et al. (2000), Breunig et al. (2001)] may overestimate actual distance between two groups of data points (data bubbles). Because, data bubbles

in [Breunig et al. (2000), Breunig et al. (2001)] do not account patterns outside hyper-sphere of radius *extent* centered at centroid while calculating distance between data bubbles. However, these outside data points play important role in measuring distance between a pair of clusters in single-link method and help single-link detect non-convexed clusters in dataset.

- Data bubbles in [Breunig et al. (2000), Breunig et al. (2001)] assume that patterns are uniformly distributed within a bubble in order to estimate K-NN distance of the representative of the bubble. However, this assumption also leads to error in measuring distance between a pair of data bubbles.

- Directional data bubble [Zhou and Sander (2003)] computes distance statistics of each bubble in the direction of all other bubbles of a given dataset. Therefore, it needs to perform a large number of distance computations.

4.3 Proposed Method

This section presents a new data summarization scheme for speeding up hierachical single-link method in large dataset. Deficiency of data bubbles discussed in last subsection are addressed in the proposed scheme. The proposed *data sphere (ds)* scheme uses sequential leaders clustering method to obtain summary of a given dataset. A technique is introduced to reduce the number of distance computations in leaders clustering method. Finally, we propose to combine single-link with *data sphere* to obtain hierarchical clustering of the dataset.

4.3.1 Accelerated leader

The proposed summarization scheme utilizes leaders clustering method to collect important statistics of a dataset. One can use Modified leader proposed in Section 3.4.3 for this purpose. However, Modified leader makes a large number of unnecessary distance computations in the process of searching a suitable leader of a pattern. Modified leader

works as follows.

Let $\mathcal{L}$ be the set of leaders at an instant. For each pattern x, Modified leader computes distances between x and each successive leader in $\mathcal{L}$ until x becomes a follower or there is no suitable leader in $\mathcal{L}$. If there is no suitable leader, then x becomes a new leader and these computed distances between x and leaders happen to be very useful for constructing distance matrix of $\mathcal{L}$. However, if x becomes follower of a leader, then these distance computations turn out to be non beneficial (pointless) and Modified leader discards them. Since, number of followers is more than the number of leaders in a dataset, Modified leader frequently makes this types of pointless distance computations.

Therefore, a technique is introduced to avoid distance computations between x and certain leaders in $\mathcal{L}$ and technique defers distance computations between new leader and leaders in $\mathcal{L}$ until x is detected as a new leader. The proposed technique is termed as *Accelerated leader*, which exploits triangle inequality property of metric space. The triangle inequality property can be stated as follows.

$$\forall a, b, c \in \mathcal{D}, d(a, b) \leq d(b, c) + d(a, c) \tag{4.4}$$

where $\mathcal{D}$ is the set of data points, d is a distance function over the metric space $\mathcal{M} = (\mathcal{D}, d)$. Let l_1, l_2 be the two leaders and x be an arbitrary pattern in $\mathcal{D}$. Form equation (4.4),

$$d(x, l_2) \geq |d(l_1, l_2) - d(x, l_1)| \tag{4.5}$$

From equation (4.5) it may be noted that a lower bound on the distance between leader l_2 and pattern x (termed as $d^{lower}(x, l_2)$) can be obtained from $d(l_1, x)$ and $d(l_1, l_2)$ without calculating the exact distance between l_2 and x. This leads to the following Observation 4.1.

Observation 4.1 *Let τ be the threshold of leaders clustering method and l_1, l_2 be two leaders produced by the leaders method. If $d(x, l_1) > \tau$ and $d^{lower}(x, l_2) > \tau$, then x can neither be follower of l_1 nor l_2.*

Observation 4.1 states that having known the distance $d(l_1, l_2)$ between a pair of leaders

l_1, l_2, distance $d(x, l_1)$ between a pattern x, l_1, one can surely conclude that x cannot be follower of leader l_2, if $|d(l_1, l_2) - d(x, l_1)| > \tau$. Therefore, actual distance computation between x and leader l_2 can be avoided.

Algorithm 4.1 Accelerated leader($\mathcal{D}$, τ)

{$\mathcal{D}$: Dataset and τ : leaders threshold distance }
$\mathcal{L} \leftarrow \{l_1\}$;
$D^l[\,][\,] \leftarrow 0$; { $D^l[\,][\,]$ is Distance Matrix for leaders }
for each $x \in \mathcal{D} \setminus l_1$ **do**
  $S \leftarrow \mathcal{L}$; $MIN = \infty$;
  $T[\,|\mathcal{L}|\,] \leftarrow 0$ {$T[\,.]$ is a temporary array}
  while (x does not become a follower and S is not empty) **do**
    Pick a leader l_i and delete from S. //* l_i is earliest generated leader among the leaders in S *//
    if $d(x, l_i) \leq \tau$ **then**
      x becomes a follower of l_i; break;
    else if $d(x, l_i) < MIN$ **then**
      $MIN = d(x, l_i)$;
      $T[i] = d(l_i, x)$;
      for each leader $l_k \in S (l_k \neq l_i)$ **do**
        if $d^{lower}(x, l_k) > \tau$ **then**
          delete l_k from set S.
        end if
      end for
    end if
  end while
  if (x not be follower of any existing leaders in $\mathcal{L}$) **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
    for $i = 1$ to $|\mathcal{L}|$ **do**
      if ($T[i] = 0$) **then**
        $T[i] = d(x, l_i)$
      end if
    end for
    $D^l[\,|\mathcal{L}| + 1\,][\,] \leftarrow T[\,|\mathcal{L}|\,]$
  end if
end for
Output $\mathcal{L}, D^l[\,][\,]$.

Accelerated leader works as follows. It needs a distance matrix for leaders. This distance matrix can be generated hand-in-hand during the generation of leaders (without any extra distance computation). Therefore, one can easily estimate $d^{lower}(x, l_2)$ only by computing distance $d(l_1, x)$.

Let τ be the leader's threshold. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ be the set of leaders generated at an instant. These leaders are assigned to a set L and marked them as "unprocessed" leaders. The scheme starts with calculating the distance between a new pattern x and leader l_f (where l_f is the earliest generated leader among the set of "unprocessed" leaders). If $d(x, l_f) \leq \tau$, then x becomes a follower of leader l_f. If $d(x, l_f) > \tau$, one can avoid the distance computations from all leaders $l_i \in L - \{l_f\}$ for which estimated lower bound $d^{lower}(x, l_i) > \tau$. Leaders l_i, l_f are marked as "processed" (pruned) leaders. If all leaders are pruned then x becomes a new leader and added to $\mathcal{L}$. If all leaders are not marked as "processed", the technique repeats same procedure of calculating distance between x with next unprocessed leader $l_u \in \mathcal{L}$ if $d(x, l_u) < d(x, l_f)$. If no (unprocessed) $l_u \in \mathcal{L}$ is found such that $d(x, l_u) > d(x, l_f)$, then there cannot be a leader l_j such that $d(x, l_j) \leq \tau$; so x becomes a new leader and added to $\mathcal{L}$. The steps of *Accelerated leaders* is depicted in Algorithm 4.1.

Review of related clustering approaches using Triangle inequality

Triangle inequality property of the metric space has been used to reduce the distance computations in the clustering methods [Elkan (2003), Nassar et al. (2004)]. Charles Elkan uses triangle inequality property to accelerate k-means clustering method [Elkan (2003)]. The accelerated k-means is based on the fact that most distance computations in classical k-means method are redundant and these can be avoided. The accelerated k-means repeatedly applies two lemmas (Lemma 4.1, Lemma 4.2) to avoid unnecessary distance calculations.

Lemma 4.1 *[Elkan (2003)] Let x be a pattern and let c and c' be two centers. If distance between the centers is more than the twice of the distance between the pattern x and center c, then pattern x is closer to center c compared to the center c'.*

Lemma 4.2 *[Elkan (2003)] Let x be a pattern and let c and c' be two centers. Let distance between x and c be $d(x,c)$. Then, $d(x,c') > \max\{0, d(c,c') - d(x,c)\}$, where $d(x,c')$ is the distance between x and c'.*

The accelerated k-means is shown to perform significantly less number of distance computations compared to the classical counter part. Nassar et al. exploit Lemma 4.1 to reduce a large number of distance computations in constructing incremental data bubbles in [Nassar et al. (2004)] for dynamics databases. Focus of this thesis is on static dataset. Therefore, discussion on incremental data bubble is not presented here.

4.3.2 *data sphere* :Proposed Summarization Scheme

In this section, a new summarization scheme termed as *data sphere (ds)* is introduced. Proposed summarization scheme scans a dataset twice only and does not need to keep entire dataset in the main memory of a computer system.

The proposed scheme utilizes leaders clustering method to obtain summary of the whole dataset. A *data sphere (ds)* is a summarized unit for a set of points (followers of a leader). Let τ be the leaders threshold. A *ds* is defined as follows.

Definition 4.1 (*data sphere (ds)*) *Let $X = \{x_1, x_2, \ldots, x_k\} \subseteq \mathcal{D}$ be the followers of a leader l. A ds for X is defined as a 4-tuple $ds = (k, l, \mu + \alpha\sigma, P)$, where*
μ = the average distance from leader l to its followers,
σ = standard deviation of distances from leader to its followers,
$\alpha \in \mathbb{R}$, $0 < \alpha \leq (\tau - \mu)/\sigma$;
P = a subset of followers of l, which lie outside the hyper-sphere of radius $\mu + \alpha\sigma$ centered at l. i.e. $P = \{x_i \in X \mid ||l - x_i|| > \mu + \alpha\sigma\}$ □

Classical leaders clustering method outputs only set of leaders. Therefore, it cannot be used directly to collect information of followers of leaders. Leaders clustering method is modified in order to stores statistics of its followers of each leader l in the form $S^l = (k, l, ld, sd)$, where k = number of followers including the leader l, $ld = \sum_{i=1}^{k} d_i$, $sd = \sum_{i=1}^{k} d_i^2$. d_i is the distance from the l to its i^{th} follower, *i.e.* $d_i = ||l - x_i||$, x_i is the i^{th}

follower of leader l. Once all parameters (k, l, ld, sd) of a *data sphere* are collected, μ and σ of the corresponding *data sphere* (leader) are calculated as follow.

$$\mu = \frac{ld}{k}; \sigma = \sqrt{(sd/k) - \mu^2}$$

Chebyshev's inequality (Theorem 4.2) is used to find the value of "α" of a *data sphere*. For completeness, the theorem is reproduced here.

Theorem 4.2 *If t is a random variable with mean and standard variation μ_t and σ_t, respectively, then for any positive real number α, the probability*
prob $\{t \geq (\mu_t + \alpha\sigma_t)\} \leq 1/(1 + \alpha^2)$. □

As d_i is a random variable and for a given probability *prob*, value of α can be computed using Theorem 4.2 for each *data sphere*. It is noted that scheme does not assume nature of distribution of followers. Chebyshev's inequality is independent of any distribution.

To identify followers of a leader l which lie outside the hyper-sphere of radius $\mu + \alpha\sigma$ centered at l (P of a *ds* corresponding l), leaders method is applied once more in the same order with same set of leaders $\mathcal{L}$ and same threshold distance τ as follows.

For each pattern $x \in \mathcal{D}$, leaders method starts searching a suitable leader $l \in \mathcal{L}$. The searching process begins from earliest generated leader in $\mathcal{L}$. The pattern x is included as a member of P of a *data sphere* corresponding to the leader l, if it holds the following condition.

$$\mu_l + \alpha_l \times \sigma_l \leq ||x - l|| \leq \tau$$

where μ_l is the average distance from l to its followers and σ_l is the standard deviation of distances from l to its followers. The summarization scheme is depicted in Algorithm 4.2.

Each *data sphere (ds)* is a compressed representation of followers of a leader. It also be noted that followers of a leader l, are also members of the corresponding *data sphere*. All *data sphere*s form a compressed (summarized) set $\mathcal{S}$ of the dataset. The *data sphere* summarization scheme can be applied in any metric space (vector or non vector).

Time and space complexity are analyzed as follows.

Algorithm 4.2 *data-sphere* summarization $(\mathcal{D}, \tau)$

Apply leaders clustering method to compute (k, l, ld, sd) for each leader l.
Compute μ, σ and α for each *data sphere*.
for each data sphere ds_l **do**
  $P_{ds_l} = \emptyset$ $\quad / * ds_l = (k_l, l, \mu_l + \alpha_l\sigma_l, P_{ds_l}) * /$
end for
/* leaders method is applied for second time*/
for each $x \in \mathcal{D}$ **do**
  Find a suitable leader l from leaders set /* leaders are searched according to generation time. */
  if $(x == l)$ **then**
    Continue;
  else
    if $(||x - l|| \geq \mu_l + \alpha_l\sigma_l)$ **then**
      $P_{ds_l} = P_{ds_l} \cup \{x\}$
    end if
  end if
end for
Let $\mathcal{S}$ be the set of all *data sphere*s.
Output $\mathcal{S}$.

1. Step of computing (k, l, ld, sd) for all leaders takes $O(mn)$, where m is the number of leaders in the dataset. Space requirement is $O(m)$.

2. μ, σ and α for each *data sphere* can be calculated in constant time using the value of k, l, ld, sd.

3. Finding members in Ps of *data sphere*s takes $O(mn)$. Space complexity is $O(\sum_{i=1}^{m} |P_i|) = O(m)$.

Overall time complexity of the proposed summarization scheme is $O(mn)$ and space complexity is $O(m)$.

4.3.3 Proposed Clustering Method

The summarized set $(\mathcal{S})$ of dataset $\mathcal{D}$ produced by *data sphere* discussed in previous subsection is used for clustering large dataset. The agglomerative single-link method is applied to compressed set $\mathcal{S}$ to obtain hierarchical clustering of the entire dataset. The

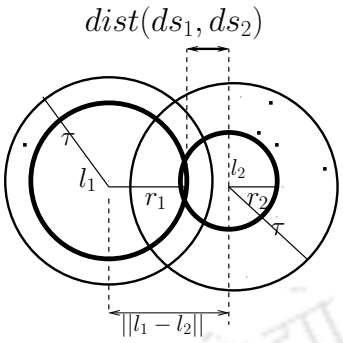


Figure 4.3: Distance between a pair of intersecting *data spheres*.

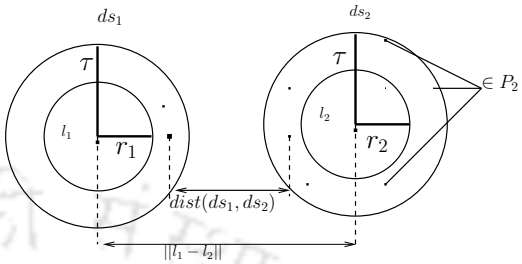


Figure 4.4: Distance between two non-intersecting *data spheres*.

proposed clustering method is termed as *summarized single-link (SSL)* method. A suitable distance/ similarity measure between a pair of *data sphere* needs to be defined in order to apply SSL method to the set of *data spheres* ($\mathcal{S}$). Distance between two representatives (leaders) of a pair of *data spheres* does not reflect effective distance between the pair of *data spheres*. In accordance with the distance measure in single-link clustering method[1], an effective distance between a pair of *data spheres* is defined as follows (Definition 4.2).

Definition 4.2 (Distance between a pair of *data spheres*) *Let* $ds_1 = (k_1, l_1, r_1, P_1)$ *and* $ds_2 = (k_2, l_2, r_2, P_2)$, *where* $r_1 = \mu_1 + \alpha_1\sigma_1$, $r_2 = \mu_2 + \alpha_2\sigma_2$ *be two data spheres. Then, distance between* ds_1 *and* ds_2 *is defined as follows.*

$$dist(ds_1, ds_2) = \begin{cases} \min\left(dis_{out}(P_1, P_2), ||l_1 - l_2|| - (r_1 + r_2)\right) \\ \qquad \textit{if}(||l_1 - l_2|| - (r_1 + r_2)) \geq 0 \\ ||l_1 - l_2|| - \max\left(r_1, r_2\right), \qquad \textit{Otherwise} \end{cases}$$

where,

$$dis_{out}(P_1, P_2) = \min\left(||x_i - x_j|| \mid x_i \in P_1, x_j \in P_2\right) \qquad \square.$$

Objective of the Definition 4.2 is to find distance between two closest patterns of two *data spheres*. Two situations appear while calculating an effective distance between a pair of *data spheres*.

[1]Let C_1 and C_2 be two groups (clusters) of patterns, respectively. Then, distance between them: $D_{average}(C_1, C_2) = \min\{||x_i - x_j|| \mid x_i \in C_1, x_j \in C_2\}$

- If inner hyper-sphere of two *data spheres* intersects each other, then the effective distance is the distance between two leaders minus maximum of radii of two inner hyper-spheres (Figure 4.3).

- For non-intersecting *data spheres*, effective distance is calculated as follows. Let distance between two leaders minus radii of their inner hyper-spheres (Figure(4.4)) be dis_{in} and let minimum distance between the patterns in P_1 and P_2 be dis_{out}. Finally, effective distance is measured by taking the minimum of dis_{in} and dis_{out}.

Effective distances between all pairs of *data spheres* are calculated as per Definition 4.2. Initially, each *data sphere* is treated as a cluster. Iteratively, the SSL method merges closest cluster pairs and updates effective distance matrix. The process of merging of clusters and updation continues until number of cluster is one. Finally, each *data sphere* is replaced by members of the corresponding *data sphere*. The SSL method is shown in Algorithm 4.3.

Algorithm 4.3 Summarized-single-link clustering $(\mathcal{D}, \tau)$

Apply *data-sphere* summarization $(\mathcal{D}, \tau)$ as described in Algorithm 4.2
/*Let $\mathcal{S}$ be the output of *data-sphere* summarization(.) */
Apply agglomerative single-link method to $\mathcal{S}$ with effective distance measure given Definition 4.2.
Let results be $\pi_{\mathcal{S}} = \{\pi_1, \pi_2, \ldots\ldots, \pi_{|\mathcal{S}|}\}$, set of partitions of $\mathcal{S}$.
Replace each $ds_l \in \pi_i$ by members (followers) of ds_l (leader l). Let $\pi_{\mathcal{D}} = \{\pi_{D_1}, \pi_{D_2}, \ldots\ldots, \pi_{D_{|\mathcal{S}|}}\}$ be the output.
Output $\pi_{\mathcal{D}}$.

4.3.4 Complexity Analysis

The complexity of SSL is analyzed as follows.

1. The summarization scheme has time complexity of $O(mn)$

2. To compute effective distance between a pairs of data spheres ds_{l_1} and ds_{l_2}, method calculates distance between members of P_1 and P_2. Generally, cardinality of P_i is very less compared to the number of *data spheres* (leaders). So, step of calculating effective distance between all pair of data spheres takes time of $O(m^2)$.

Table 4.1: Dataset used in experiments

Dataset	Pattern#	Feature#	Class#
Banana	4900	2	2
Pendigits	7494	16	10
GDS10	23709	28	2
Shuttle	58000	9	7
Spiral(Synthetic)	3330	2	2
Circle(Synthetic)	28000	2	4

3. Single-link($\mathcal{S}$) has time and space complexity of $O(m^2)$.

Overall time complexity of SSL method is $O(mn) + O(m^2) = O(mn)$. The space complexity is (m^2). The SSL scans dataset twice.

4.4 Experimental Results

Performance of proposed distance reduction technique and clustering method SSL are experimentally evaluated in this section. First reduction technique is evaluated on one synthetic and one real world dataset. Subsequently, effectiveness of the SSL is tested. Two synthetic and four real world dataset (`http://archive.ics.uci.edu/ml/dataset.html`) are used in our experiments. Spiral and Circle are 2-dimensional synthetic dataset. The Spiral data has two spiral shaped clusters with 3330 patterns. Circle has well separated four clusters of circular shapes. Brief description of the dataset are given in Table 4.1.

4.4.1 Performance of Accelerated leaders

Classical Leaders, Modified leader (proposed in Section 3.4.3) clustering and *Accelerated leader* are implemented using C language and executed on Intel Core 2 Duo CPU (3.0 GHZ) with 2GB RAM IBM PC. These three methods are tested with Circle and Shuttle dataset. The detailed results are shown in Table 4.2 and Figure 4.5. It may be noted that all three leaders generate distance matrix along with a set of leaders. Number of distance computations performed by classical leaders in the experiments include the

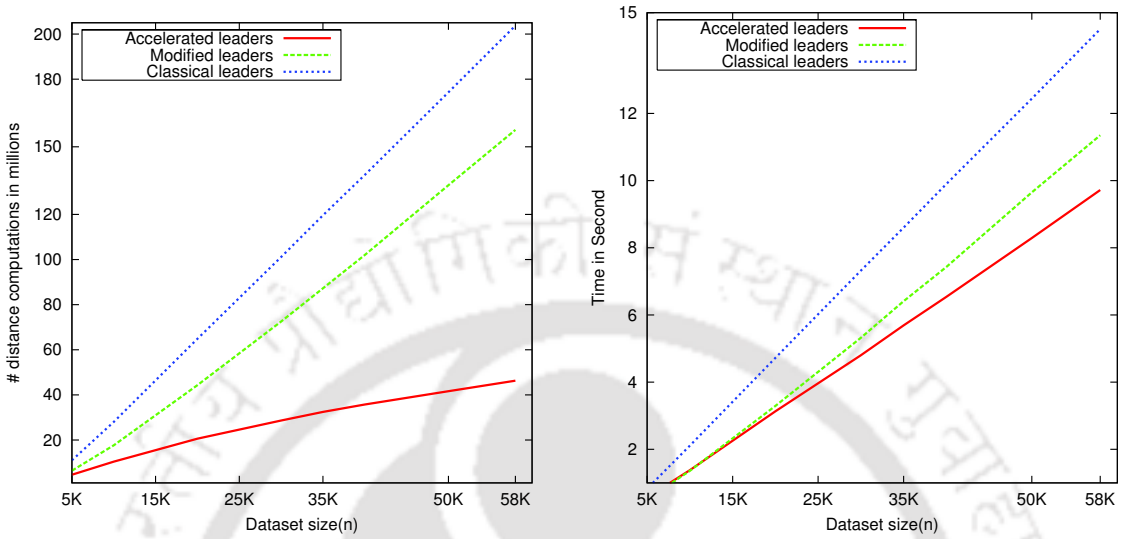
Table 4.2: Number of Distance Computations with Circle dataset

Threshold (τ)	Method	# Computations (in Million)
0.01	Classical leaders	307.03
	Modified leaders	189.79
	Accelerated leader	**117.27**
0.1	Classical leaders	118.57
	Modified leaders	90.13
	Accelerated leader	**28.66**
0.2	Classical leaders	22.13
	Modified leaders	20.03
	Accelerated leader	**2.37**
0.3	Classical leaders	12.04
	Modified leaders	11.33
	Accelerated leader	**0.96**
0.4	Classical Leaders	6.20
	Modified Leaders	5.97
	Accelerated leader	**0.49**

cost of constructing the distance matrix. The value of the τ are determined based on intended compression ratio of the proposed summarization scheme, which is discussed in next subsection.

Proposed *Accelerated leader* performs significantly less computations compared to that of the classical leaders method (Table 4.2). Experiments with Circle dataset shows that *Accelerated leader* makes 180 millions less distance computations to achieve same results as that of the classical leaders method (Table 4.2) with $\tau = 0.01$ (number of leaders=15313). The Accelerated leader performs 70 millions less distance computations compared to Modified leaders with $\tau = 0.01$. Similarly, reduction technique avoids a large number of distance calculations with different values of τ (Table 4.2). With $\tau = 0.4$ (number of leaders= 675), Accelerated method has more than one order speed up factor in terms of number of distance computations.

To show the performance of the proposed reduction technique with variable dataset size, experiments are conducted on Shuttle dataset with leaders threshold $\tau = 0.001$. This is shown in Figure 4.5. It may be noted that with increase of the dataset size, number of



(a) Distance computations Vs dataset size (b) Execution time Vs dataset size

Figure 4.5: Performance of *Accelerated leader* with Shuttle dataset ($\tau = 0.001$)

distance calculations reduces significantly compared to the classical leaders and Modified leaders (Figure 4.5(a)).

Figure 4.5(b) shows execution time of proposed distance reduction technique with varying dataset size. It may be noted that with the increase of dataset size, execution time of accelerated method improves considerably compared to the other approaches. This shows that accelerated method is more scalable with the size of the dataset compared to the other leaders methods.

4.4.2 Performance of summarized single-link method

To show the effectiveness of the proposed *data sphere* scheme over *Data Bubble* summarization, classical single-link, single-link with data bubble ($dbSL$), single-link with directional data bubble ($\overrightarrow{db}SL$) and proposed SSL methods are implemented on Intel Core 2 Duo CPU (3.0 GHZ) with 2GB RAM IBM PC.

Rand Index (RI)[Rand (1971)] and Purity [Zhao and Karypis (2002)] are used to measure clustering similarity (closeness) between a pair of clusterings of a dataset. Given

a dataset $\mathcal{D}$ of n patterns and two clusterings π_1 and π_2 of $\mathcal{D}$, *Purity* is defined as follows.

$$Purity(\pi_1, \pi_2) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\pi_2|} \max_j | \ C_i^{(2)} \cap C_j^{(1)} \ |$$

where $C_i^{(2)} \in \pi_2$ and $C_j^{(1)} \in \pi_1$.

Data summarization technique forms a compressed representative set of a given dataset. A data summarization technique is oftenly characterized by a factor called *Compression Ratio (CR)*. The *Compression Ratio (CR)* is defined as ratio of the size of a given dataset to the size of compressed representative set. The CR for Data Bubbles is defined in equation (4.6).

$$CR = \frac{n}{\text{Number of data bubbles}} \tag{4.6}$$

Similarly, CR for *data sphere* is given in equation (4.7). It may be noted that denominator of equation (4.7) has two parts. First part ($|\mathcal{S}|$) is determined by leaders threshold τ and $|P_i|$ in second part is the number of patterns outside inner sphere of data sphere ds_i (Figure 4.3, Figure 4.4). The value of $|P_i|$ in a *data sphere* is calculated using Chebyshev's inequality (Theorem 4.2). For performance analysis with Data Bubble, a suitable value of τ and *prob* in Theorem 4.2 are selected to match the values of denominators of equation (4.6) and equation (4.7).

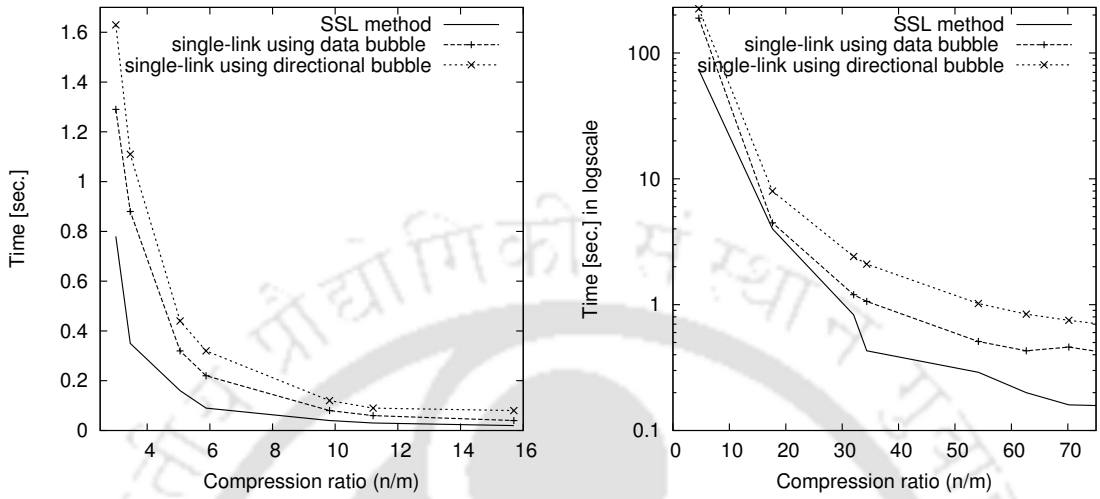
$$CR = \frac{n}{|\mathcal{S}| + \sum_{i=1}^{|\mathcal{S}|} |P_i|} \tag{4.7}$$

To evaluate results of various clustering methods, a clustering is considered as the final output produced by a method in which number of clusters is same as the number of classes in a given dataset. *RI* and *Purity* (Table 4.3) are computed between the clusterings produced by classical single-link and each of the three methods SSL, *dbSL*, and $\overrightarrow{dbSL}$ separately. The value of *prob* in SSL method is considered as 0.1 for Spiral, Banana and Pendigits dataset.

Table 4.3: Results for different dataset

Dataset	CR	Method	Time (in sec.)	Rand Index (RI)	Purity
Spiral	17.61	SSL	0.01	1.000	1.000
	17.61	*dbsl*	0.02	0.891	0.981
	17.61	$\overrightarrow{db}SL$	0.05	0.996	0.999
	–	Single-link	22.12	–	–
Banana	20.85	SSL	0.03	1.000	1.000
	20.85	dbSL	0.05	0.998	0.999
	20.85	$\overrightarrow{db}SL$	0.09	0.818	0.917
	–	Single-link	43.12	–	–
	41.17	SSL	0.02	1.000	1.000
	41.17	dbSL	0.02	0.997	0.999
	41.17	$\overrightarrow{db}SL$	0.05	0.818	0.917
	–	Single-link	43.12	–	–
Pendigits	5.50	SSL	1.07	0.984	0.999
	5.50	dbSL	3.66	0.981	0.998
	5.50	$\overrightarrow{db}SL$	4.57	0.985	0.998
	–	single-link	512.78	–	–
	8.50	SSL	0.31	0.984	0.999
	8.50	dbSL	1.38	0.980	0.998
	8.50	$\overrightarrow{db}SL$	1.80	0.984	0.999
	–	single-link	512.78	–	–
	17.07	SSL	0.11	0.983	0.999
	17.07	dbSL	0.53	0.977	0.997
	17.07	$\overrightarrow{db}SL$	0.66	0.979	0.990
	–	single-link	512.78	–	–

With $CR = 17.61$, the SSL method produces exactly same clustering output as produced by the classical single-link method for **Spiral** dataset (Table 4.3). However, proposed SSL method is three order faster than the single-link method applied directly to the dataset. The single-link method with data bubble [Breunig et al. (2001)] cannot produce exactly same clustering results with same compression ratio for Spiral data. However, clustering outputs produced by single-link with directional data bubble [Zhou and Sander (2003)] are very close ($RI = 0.996$) to the output produced by the classical method. The single-link with *data sphere* is the fastest one (Table 4.3).



(a) Experiments with *Spiral* dataset (b) Experiments with *Circle* dataset

Figure 4.6: Execution time taken by *summarized single-link* , *single-link with data bubble* [Breunig et al. (2001)] and *single-link with directional data bubble* [Zhou and Sander (2003)] methods as compression ratio varies for two synthetic dataset.

Experiment with **Banana dataset** shows that proposed SSL can find exactly same clustering results ($RI = 1.000$) as that of the single-link method with high compression ratio $CR = 41.17$. However, it is more than three order time faster than the classical single-link method. Proposed summarization scheme outperforms both approaches of data bubble summarization schemes with $CR = 20.85$ and $CR = 41.17$ (Table 4.3).

Time taken by SSL, *dbSL* and $\overrightarrow{dbSL}$ methods as CR varies is shown in Figure 4.6(a) and Figure 4.6(b) for the **Spiral** and **Circle** dataset, respectively.

The time taken by the SSL, dbSL, $\overrightarrow{dbSL}$ and single-link method as the dataset size increases for a fixed compression ratio ($CR = 9.35$) is shown in Figure 4.7 for the *Circle* dataset. The plot is semi-log. The SSL method is significantly faster than that of the classical single-link method. The plot also shows that the proposed *data sphere* summarization scheme outperforms other variant of Data Bubble summarization scheme at execution time.

The SSL, dbSL and $\overrightarrow{dbSL}$ methods generate three different clustering hierarchies for a given dataset. Rand Index at different levels of the hierarchy produced by the SSL, dbSL

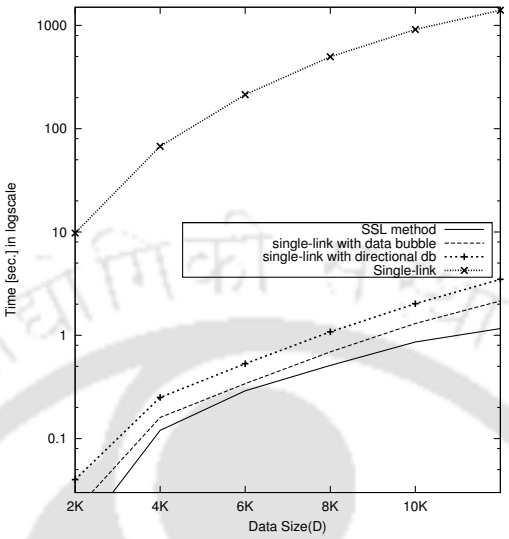


Figure 4.7: Time taken by SSL, *dbSL*, single-link with directional data bubble and classical single-link methods as dataset size varies for **Circle** dataset.

and $\overrightarrow{db}SL$ method with respective single-link method for **Circle** dataset are computed at $CR = 260$ (Table 4.4). The clustering results of SSL are very close to that of the classical single-link method. It outperforms dbSL at RI and Purity measures. The clustering outputs are close to the outputs produced by $\overrightarrow{db}SL$ method.

Experiments with large dataset

For large dataset (**Circle, Shuttle, GDS10**) experiments are performed on Intel Xeon Processor (3.6GHz) with 8 GB RAM IBM server because of huge memory requirement of classical single-link method.

Type 1 diabetes gene expression data **GDS10** (`http://www.ncbi.nlm.nih.gov/geo/`) is used in experiments and results are reported in Table 4.5. The clustering results produced by the SSL method are comparable with the clustering results produced by dbSL and $\overrightarrow{db}$SL. However, it is faster than both approaches. The SSL method is more than three order faster than the classical single-link method with compression ration 10.64. Similar trends are found with compression ratio 19.23.

Shuttle dataset is also used in experimental evaluations to show the effectiveness of the

Table 4.4: Clustering hierarchy produced by SSL method with Circle dataset

# cluster	Methods	Rand Index (RI)	Purity
2	SSL	1.000	01.000
	dbSL	0.997	0.999
	$\overrightarrow{dbSL}$	1.000	1.000
3	SSL	0.627	0.716
	dbSL	0.621	0.700
	$\overrightarrow{dbSL}$	0.744	0.750
4	SSL	0.852	0.913
	dbSL	0.841	0.900
	$\overrightarrow{dbSL}$	1.000	1.000
5	SSL	0.835	0.963
	dbSL	0.836	0.964
	$\overrightarrow{dbSL}$	0.949	0.925
6	SSL	0.827	0.897
	dbSL	0.822	0.888
	$\overrightarrow{dbSL}$	0.921	0.876
7	SSL	0.813	0.860
	dbSL	0.800	0.850
	$\overrightarrow{dbSL}$	0.892	0.839

proposed SSL method. The SSL method is faster than both approaches of data bubbles and it outperforms dbSL method in clustering quality. The SSL method is more than three order faster than the classical single-link method with compression ration 18.70.

4.5 Conclusion

Hierarchical clustering methods like single-link can find arbitrary shaped clusters with nested structures in a dataset. These methods are not scalable with the size of the dataset and scans the dataset many times. Therefore, these methods are not suitable for clusters analysis in large data. In this chapter, a summarization scheme *data sphere* is proposed to speed up classical single-link method. The single-link with *data sphere* method takes considerably less time compared to the classical single-link method. It

Table 4.5: Results for different dataset

Dataset	CR	Method	Time (in sec.)	Rand Index (RI)	Purity
GDS10	10.64	SSL	7.68	0.818	0.917
	10.64	dbSL	21.15	0.801	0.892
	10.64	$\overrightarrow{db}SL$	26.33	0.887	0.961
	–	single-link	4215.51	–	–
	19.23	SSL	1.88	0.801	0.892
	19.23	dbSL	7.93	0.711	0.815
	19.23	$\overrightarrow{db}SL$	9.88	0.881	0.955
	–	single-link	4215.51	–	–
Shuttle	18.7	SSL	30.12	0.740	0.749
(40,000)	18.7	dbSL	37.70	0.731	0.748
	18.7	$\overrightarrow{db}SL$	53.40	0.819	0.918
	–	single-link	7101.12	–	–

scans the dataset twice. The clustering results produced by the SSL method is very close to the results produced by that of the single-link method. The SSL method outperforms single-link with Data Bubble [Breunig et al. (2001)] at clustering quality and execution time and outperforms single-link with directional Data Bubble [Zhou and Sander (2003)] at execution time.

The directional Data Bubble can handle "gaps" in a Data Bubble and it is found that single-link with directional Data Bubble produces clustering results very close to the classical single-link method. The next chapter proposes another summarization scheme, which is the same line of *data sphere*. However, "gaps" and distance measures between a pair of *data bubble*s are computed in a precise manner.

Chapter 5

Tolerance Rough Set Theory based data summarization for clustering large dataset

5.1 Introduction

Hierarchical clustering methods produce better clustering results than partitional clustering methods [Lin and Chen (2005)]. Main advantage of the hierarchical clustering approach over partitional clustering is that it can find inherent hierarchy/taxonomy present in data. Clustering methods like OPTICS [Ankerst et al. (1999)], single-link [Sneath and Sokal (1973)], complete-link [King (1967)], average-link [Murtagh (1984)] are typical examples of hierarchical methods. OPTICS produces density based hierarchy of clusters using two input parameters. It can find arbitrary shaped clusters. However, clustering results are influenced by the selection of parameters. The single-link, complete-link and average-link mainly differ in the "distance measure" between a pair of clusters. Complete-link and average-link are used for finding globular shaped clusters. They cannot find arbitrary shaped clusters. Single-link can find arbitrary shaped clusters and does not need any input parameter to produce hierarchy of clusters in a data. As discussed in Chapter 2, most of the existing hierarchical clustering methods are not scalable with

the size of the dataset and need multiple dataset scans to get converged. Therefore, these methods cannot be applied for cluster analysis in large dataset.

To speed up clustering activity in large dataset, data summarizations (data compression) became popular in recent years [Zhang et al. (1996), Bradley et al. (1998), Breunig et al. (2000, 2001), Zhou and Sander (2003)]. Data summarization is a scheme to obtain a representative set along with a summary of the dataset. Subsequently, a clustering method can be applied to this summary to find clustering structures of whole dataset. Data summarization schemes found in [Zhang et al. (1996), Bradley et al. (1998), DuMouchel et al. (1999)] are suitable to speed up k-means type of clustering methods.

Data Bubbles proposed in [Breunig et al. (2000, 2001), Zhou and Sander (2003)] are used to speed up density based OPTICS clustering method in large dataset. As discussed in Section 4.2 of Chapter 4, Data Bubbles cannot be used directly to speed up single-link clustering method in large dataset.

In this chapter, a new summarization scheme is proposed in order to address the drawbacks of Data Bubbles. New summarization scheme is based on tolerance rough set model (TRSM) (discussed in Section 2.3.7 of chapter 2) and termed as *rough bubble*. Proposed *rough bubble* utilizes leaders clustering method to collect sufficient statistics of the dataset and it exploits tolerance rough set to handle uncertainty associated with leaders method. The *rough bubble* is used with single-link method for cluster analysis in large dataset. Single-link with *rough bubble* clustering is termed as *rough single-link (RSL)*. The RSL method is considerably faster than classical single-link method and clusterings produced by RSL are close to the clustering produced by single-link method. The RSL outperforms directional data bubble [Zhou and Sander (2003)] based single-link in clustering results and execution time of RSL is marginally higher than the directional bubble based single-link method.

Contributions made in this chapter are summarized as follows.

- Tolerance Rough Set Model (TRSM) based new data summarization scheme termed *rough bubble* is proposed to work with single-link method in large dataset.

- Proposed *rough bubble* can handle "gaps" (cluster separation) appeared in a *bubble* more efficiently than the approached used in [Zhou and Sander (2003)] for data bubbles.

- The *rough bubble* has two steps. In the first step, representatives (leaders) are detected by leaders method and directional distance statistics are collected in the last step. A technique is devised to speed up first step (leaders method). Proposed technique termed *TI-LEADER* utilizes triangle inequality property to avoid unnecessary distance computations. Unlike Accelerated leaders (proposed in Chapter 4), TI-LEADER does not need distance matrix.

- Clustering results produced by *RSL* method on synthetic and real dataset are analyzed using *Rand Index and Purity*.

Rest of the chapter is organized as follows. Section 5.2 describes a brief summary of related research work. Proposed distance reduction technique and summarization scheme are described in Section 5.3. Experimental evaluations and conclusion of this chapter are discussed in Section 5.4 and 5.5, respectively.

5.2 Related work

Popular data summarization schemes namely Clustering Feature [Zhang et al. (1996)], Data squashing [DuMouchel et al. (1999)], Data Bubble [Breunig et al. (2000), Breunig et al. (2001)], directional Data Bubble [Zhou and Sander (2003)] have already been discussed in Section 4.2 of Chapter 4, elaborately. Therefore, discussion on those schemes are not included in this chapter. However, following points are to be noted for directional Data Bubble [Zhou and Sander (2003)].

- Formation of data bubbles are dependent on the initial selection of patterns (seed patterns), which are randomly picked.

- Lost of cluster separation ("gap") is handled using only distance information between patterns to their representatives.

- This scheme may detect "gap" in a data bubble (say O_1). However, patterns in O_1 are not reassigned to proper data bubbles. This step is necessary to obtain proper cluster structures.

A brief summary of clustering methods which use tolerance rough set model is presented here (detailed discussion provided in Subsection 2.3.7 of Chapter 2).

- **Document clustering based on TRSM [Ho and Nguyen (2002)]:** In this approach, tolerance relation plays a major role in finding similarity between a pair of documents. Similarity between a pair of documents is the function of upper approximations of the index terms present in each of the documents. Upper approximation of $X \subseteq \mathcal{T}$ is $\overline{T}(X)$:

$$\overline{T}(X) = \{t_i \in \mathcal{T} : T(t_i) \cap X \neq \emptyset\} \tag{5.1}$$

  where $\mathcal{T}$ is the set of index terms in document set and $T(t_i)$ is the tolerance class of an index term t_i. Having calculated upper approximations of all documents, method selects k representative documents randomly. A document x can be assigned to multiple representatives if similarity between x and each representative is more than a pre-specified parameter. It again selects representative from each cluster and repeats the process of assignment of patterns to representatives.

  S. Kawasaki et al.[Kawasaki et al. (2000)] exploit tolerance rough set model to enrich document representation in terms of semantics relatedness between documents. Weights of a index term in a document is calculated based on the upper approximation of the document. Method demonstrates that TRSM based document representation outperforms VSM (vector-space-model) based representation for the retrieval purpose. This TRSM is useful for document clustering as tolerance class is defined based on co-occurrence of index terms in documents set.

- **Clustering web transactions using rough approximation [De and Krishna (2004)] :** In this method, similarity between two transactions is measured using *Similarity Upper Approximation*, which is the function of successive upper approximations of a set of transactions X. Similarity Upper Approximation is calculated

for each transaction in the dataset. Finally, transactions are merged into a cluster with same similarity upper approximation value. The method has time complexity of $O(n^2)$, where n is the number of transactions.

- **Hierarchical clustering method based on TRSM [Kumar et al. (2007)]:** This methods accounts number of shared elements between two successive upper approximations of a transaction while calculating Similarity Upper Approximation of each transaction. Let $\overline{T}(x)$ be the first upper approximation of x. Then, second upper approximation of x is :

$$\overline{T}\ \overline{T}(x) = \{x_j \in \bigcup_{x_l \in \overline{T}(x)} \overline{T}(x_l) : \text{RelSim}(x, x_j) \geq \sigma\},$$

where $RelSim(x, x_j) = \dfrac{|\overline{T}(x) \cap \overline{T}(x_j)|}{|\overline{T}(x) - \overline{T}(x_j)|}$ Let S_i be the upper approximation of x_i, method calculates next higher order approximations S_i'. If $S_i = S_i'$, then transactions in S_i' are merged into a cluster, otherwise method repeats the process of calculating next higher order approximations. They showed that their method outperforms classical complete-link clustering method.

5.3 The Proposed Method

As discussed in Section 4.2 of Chapter 4 and Section 5.2 of this chapter, data summarization is a potential contender for clustering large size dataset. In this section, a new summarization scheme called *rough bubble* is proposed. Subsequently, single-link method is applied to *rough bubble* with appropriate distance measure to obtain clustering hierarchy. Proposed summarization scheme exploits leaders clusters method and tolerance rough set model. To speed up the *rough bubble* construction process, a technique is introduced next.

5.3.1 TI-LEADER: Speeding up leaders clustering method by means of Triangle Inequality

Proposed *rough bubble* summarization scheme initially detects representatives of a set of rough bubbles (leaders) utilizing leaders clustering method. Accelerated leader (Section 4.3.1 of Chapter 4) can be used to find the set of leaders (bubble representatives) to avoid a large number of distance computations. However, Accelerated leader method needs to compute distance matrix for leaders along with the leaders set. Computation costs for constructing distance matrix at this stage is not worthy as summarization scheme needs to compute distance from a pattern to each leader in the next step for finding directional statistics of a leader.

To avoid unnecessary distance computations in the first step of *rough bubble*, a technique is introduced, which does not compute distance matrix. Proposed distance reduction technique is termed as Triangle-Inequality based Leaders method (*TI-LEADER*). Proposed technique borrows the idea given in [Kryszkiewicz and Lasek (2010b)], which was used to speed up DBSCAN method. Proposed technique *TI-LEADER* uses Corollary 5.2, which is derived from Theorem 5.1 [Kryszkiewicz and Lasek (2010b)].

Theorem 5.1 *[Kryszkiewicz and Lasek (2010b)] Let D be an ordered list of n datapoints arranged in non-decreasing way with respect to their distances to $\overrightarrow{0}$ (origin). Let p be a point in D, q_f be a succeeding point of p such that $||q_f|| - ||p|| > EPS$, and q_b be a preceding point of p such that $||p|| - ||q_b|| > EPS$, Then:*

- *Distance between p and q_f is more than EPS. Distance between p and any succeeding points of q_f in D is more than EPS.*
- *Distance between p and q_b is more than EPS. Distance between p and any preceding point of q_b is more than EPS.*

where EPS is a positive real number and $d(.)$ is a distance metric. □

Theorem 5.1 states that one can avoid distance computations between point p and all succeeding points of q_f, between p and all preceding points of q_b while finding EPS

neighbor of p. Following Corollary 5.2 is derived to restrict search space for finding a suitable leader.

Corollary 5.2 *Let $\mathcal{L}$ be the set of leaders generated by the leaders clustering method with threshold τ at an instance. Let L be a sorted list of leaders and arranged them in non-decreasing way with respect to their distances to $\overrightarrow{0}$ (origin), where distance function is a metric. A point $x \in \mathcal{D}$ $(x \notin \mathcal{L})$ is inserted into the list L such that L remains sorted. Let l_f be first leader following x in L such that $||l_f|| - ||x|| > \tau$, and l_b be first preceding leader of x in L, such that $||x|| - ||l_b|| > \tau$. Then :*

- *x can neither be a follower of leader l_f nor any leader following l_f in L.*
- *x can neither be a follower of leader l_b nor any leader preceding l_b in L.*

Proof: Let d be the distance function. By hypothesis,

$$||l_f|| - ||x|| > \tau, \quad \text{or,} \quad d(l_f, \overrightarrow{0}) - d(x, \overrightarrow{0}) > \tau. \tag{5.2}$$

Again using triangle inequality we can write,

$$d(x, l_f) + d(x, \overrightarrow{0}) \geq d(l_f, \overrightarrow{0}) \quad \text{or,} \quad d(x, l_f) \geq d(l_f, \overrightarrow{0}) - d(x, \overrightarrow{0}) \tag{5.3}$$

From equation (5.2) and equation (5.3), we can say that

$$d(x, l_f) > \tau \tag{5.4}$$

Therefore, x cannot be a follower of leader l_f. Equation (5.2) holds for any succeeding leaders of l_f. So, equation (5.4) is valid for any succeeding leader of l_f, i.e. x cannot be the follower of any succeeding leader of l_f in L.

Analogously, it can be show that x can neither be follower of l_b or preceding leaders of l_b in L. □

Corollary 5.2 states that one can avoid distance computations between x and l_f, x and

any leaders following l_f in L. Similarly, distance computations between x and l_b, x and any leader preceding l_b in L can be avoided. To find a leader of x, TI-LEADER calculates distances only between x and the leaders which are listed between l_b and l_f in the sorted list L. The *TI-LEADER* works as follows.

Let τ be the leaders threshold and $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ be the set of leaders generated at an instance. L is a sorted list of leaders which are arranged with respect to their distances to $\overrightarrow{0}$. The technique starts calculating distance between a new pattern x and $\overrightarrow{0}$ ($||x||$), and inserts x into a proper position in the list such that the list remains sorted. It collects probable leaders of x and insert into another list LIST starting from closest preceding leader of x in the list L. *TI-LEADER* continues collecting preceding leaders until it finds a leader l_b such that $||x|| - ||l_b|| > \tau$ or it encounters the very first leader in L. Similarly, it inserts succeeding leaders into the LIST until it finds a leader l_f such that $||l_f|| - ||x|| > \tau$ or last leader is encounter in L. Finally, the technique calculates actual distances between x and each leader l_p in the LIST. If there are more than one leaders l_p such that $d(x, l_p) \leq \tau$, it picks earliest generated leader as the leader of x and removes x from the sorted list L. Otherwise, if there is no leader such that $d(x, l_p \mid l_p \in LIST) \leq \tau$, then x becomes a new leader and added to $\mathcal{L}$. This process continues till there is a pattern in the dataset. The technique is depicted in Algorithm 5.1.

Time and space complexity of the reduction technique are analyzed as follow.

1. Step of inserting an element into the sorted list T takes $O(m)$, where m is the number of leaders. This step takes $O(mn)$ for all patterns in the dataset. The space complexity is $O(m)$.

2. To generate *LEADER SET*, technique searches preceding and succeeding leaders in the list L. This step takes time of $O(m)$. Finally, it calculates distance from a pattern to all leaders in *LEADER SET*. For all patterns it takes $O(mn)$.

Overall time and space complexity of the *TI-LEADER* is $O(mn) + O(mn) = O(mn)$. The space complexity is $O(m)$. TI-LEADER is used in the first phase of the proposed summarization scheme, which is presented next.

Algorithm 5.1 *TI-LEADER*($\mathcal{D}$, τ)

$\mathcal{L} \leftarrow \{l_1\}$ { l_1 be the first pattern in $\mathcal{D}$ }
L is a sorted list of leaders in $\mathcal{L}$ with respect to $|\,|\,.\,|\,|$
for each pattern $x \in \mathcal{D} \setminus \{l_1\}$ **do**
  Insert x into the sorted list L such that L remains sorted one with respect to $|\,|\,.\,|\,|$

  LEADER SET = $\emptyset$; /* Temporary array to collect probable leaders */
  Let p_i be the closest leader preceding x in list L.
  while $(||x|| - ||p_i|| \leq \tau)$ **do**
    LEADER SET= LEADER SET $\cup\{p_i\}$.
  end while
  Let q_i be the closest pattern succeeding x in list L
  while $(||x|| - ||q_i|| \leq \tau)$ **do**
    LEADER SET = LEADER SET $\cup\{q_i\}$
  end while

  for each leader $l_j \in$ LEADER SET **do**
    if $(|\,|\,x - l_j\,|\,| > \tau)$ **then**
      Delete l_j from LEADER SET.
    end if
  end for
  if {LEADER SET == $\emptyset$} **then**
    x becomes new leader and added to $\mathcal{L}$.
  else
    x becomes follower of first generated leader in LEADER SET.
    Remove x from L
  end if
end for
Output $\mathcal{L}$

5.3.2 Summarization scheme

The proposed summarization scheme utilizes classical leaders clustering method to acquire distance statistics of each leader obtained in first phase. Tolerance rough set theory is exploited to resolve the uncertainty associated with leaders method.

Let τ be the leaders threshold distance and $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ be the set of all leaders of a dataset $\mathcal{D}$. For a pattern x and leaders l_1, l_2, it may be observed a scenario such that $||l_1 - x|| <= \tau, \quad ||l_2 - x|| <= \tau$. Then x can be eligible to be a follower of both leaders. However, classical leaders clustering method assigns x to a leader which one is observed

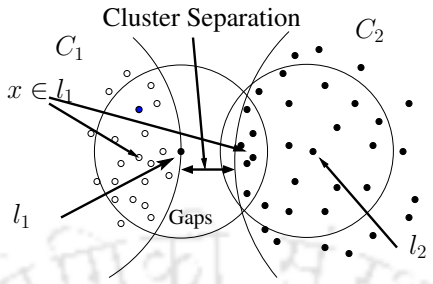


Figure 5.1: Leader l_1 contains "gap" in the direction of leader l_2

first (say l_1). It may happen that pattern x and leader l_2 belong to a cluster C_2 in a dataset, whereas l_1 belongs to a different cluster C_1. So, leader l_1 contains patterns from both clusters (C_1, C_2) and it also contains cluster separation ("gap") (Figure 5.1). Clapping of pattern x with leader l_1 makes cluster separation invisible. This leads to error in obtaining final clustering structures in data. In proposed *rough bubble*, pattern (follower) x is allowed to be the member of both leaders (l_1, l_2) and membership uncertainty is resolved using tolerance rough set model.

Let $\mathcal{L}^r = \{l_1^r, l_2^r, \ldots, l_m^r\}$ be the set of leaders where a leader may share patterns with others leaders. We call $l_i^r \in \mathcal{L}^r$ as a *rough leader* and the $\mathcal{L}^r$ as the set of rough leaders. Note that there is a one to one correspondence between the set $\mathcal{L}$ and $\mathcal{L}^r$. In the other words, let $\{l_i\}$ and $\{l_i^r\}$ be the sets of followers of i^{th} leader $l_i \in \mathcal{L}$ and rough leader $l_i^r \in \mathcal{L}^r$, respectively. Then $\{l_i\} \subseteq \{l_i^r\}$, $l_i = l_i^r$ and $|\mathcal{L}| = |\mathcal{L}^r|$. Therefore, l_i and l_i^r are used interchangeably throughout the chapter.

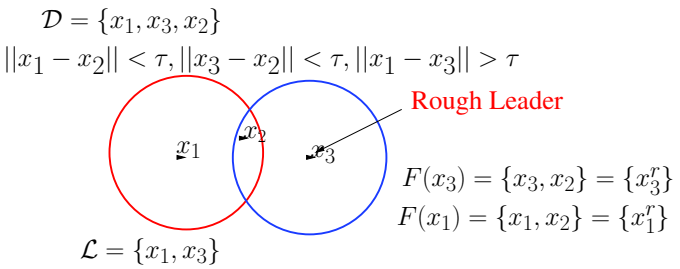


Figure 5.2: Rough leaders with follower set $F(.)$

Now, TRSM is applied to a dataset $\mathcal{D}$. Let $T \subseteq \mathcal{D} \times \mathcal{D}$ be a tolerance relation (reflexive, symmetric). One can define a tolerance class based on proximity of the patterns in $\mathcal{D}$.

Definition 5.1 (Tolerance class) *If $x \in \mathcal{D}$, then the tolerance class of x is* $T(x) = \{x_j \in \mathcal{D} \mid \quad ||x - x_j|| <= \lambda\}_{\lambda \in \mathbb{R}^+}$. □

Therefore, tolerance class of a pattern x is a set of patterns whose distance from x is less than equal to a given threshold λ. With different values of λ, we can get different tolerance relations of $\mathcal{D}$. Here, we consider the value of $\lambda = \tau/2$. One can define tolerance class and tolerant shared edge for follower set of a leader $\{l_i^r\} \subseteq \mathcal{D}$ are as follow.

Definition 5.2 (Tolerance class of a leader) *Let T be a tolerance relation on $\mathcal{D}$ and l^r be a rough leader obtained by leader threshold distance τ. The tolerance class of l^r is* $T(l^r) = \{x_j \in \mathcal{D} \mid \quad ||l^r - x_j|| \leq \tau/2\}$ □

Tolerance classes of two leaders l_1, l_2 do not intersect, *i.e.* $T(l_1^r) \cap T(l_2^r) = \emptyset$. Patterns in a tolerance class of a leader cannot be disassociated from the corresponding leader. However, patterns $x \in \{l^r\} - T(l^r)$ may need to dissociate from l^r and assign to proper leader in later stage. This leads to following definition.

Definition 5.3 (Tolerant shared edge) *Let T be a tolerance relation on $\mathcal{D}$ and l^r be a leader obtained using threshold distance τ. The tolerant shared edge of $\{l^r\}$ is defined as* $T_{SEdg}(\{l^r\}) = \{x_j \in \mathcal{D} \mid \quad \tau/2 < ||l^r - x_j|| \leq \tau\}$. □

In the proposed summarization scheme, we need to know the information of followers of each leader in the direction of all other leaders. Therefore, classical leaders method is applied to collect distance statistics of each leader. It is worthy to mention that Accelerated leader (Section 4.3.1 of Chapter 4) cannot be used for this task as it avoids distance calculation from certain leaders.

Let there be two leaders $l_1, l_2 \in \mathcal{L}$. The leader l_1 stores the information of its followers which are in the direction of l_2 (w. r. t l_2) and vice-versa. More formally :

Definition 5.4 *Let l_1 and l_2 be the two leaders. The set of followers of l_1^r in the direction of l_2^r and set of followers of l_2^r in the direction of l_1^r are $l_1^{(l_2)}$ and $l_2^{(l_1)}$, respectively, where*

Algorithm 5.2 *rough bubble* ($\mathcal{D}$, τ)

1: Apply TI-LEADER ($\mathcal{D}$, τ) as given in Algorithm 5.1
Let $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ be the output.
2: Apply classical leaders clustering method to collect statistics of each rough leader(bubble) w.r.t. all other rough leaders incrementally. /* Dataset is scanned for second time in same order */
Let $\mathbb{S} = \{S_1, S_2, \ldots, S_m\}$ be the summary of $\mathcal{D}$.
3: Output $\mathbb{S}$.

$l_1^{(l_2)} = \{x \in \{l_1^r\} : ||l_2 - x|| \leq ||l_1 - l_2||\}$,
$l_2^{(l_1)} = \{x \in \{l_2^r\} : ||l_1 - x|| \leq ||l_1 - l_2||\}$. □

Following information of a leader l_1 is stored w.r.t. leader l_2.

$$(k^{(l_2)},\ l_1,\ ld^{(l_2)},\ sd^{(l_2)},\ T(l_1),\ |T_{SEdg}|,\ |Com_{l_1}^{(l_2)}|,\ ld_{SEdg}^{(l_2)},\ sd_{SEdg}^{(l_2)})$$

where $k^{(l_2)} = |l_1^{(l_2)}|$,

$ld^{(l_2)} = \sum_{j=1}^{k^{(l_2)}} d_j$, $sd^{(l_2)} = \sum_{j=1}^{k^{(l_2)}} d_j^2$, $d_j = ||l_1 - x_j||$ and $x_j \in l_1^{(l_2)}$

$T(l_1) = \{x_j \in \mathcal{D} \mid \ ||l^r - x_j|| \leq \tau/2\}$

$T_{SEdg}^{(l_2)}(\{l_1^r\}) = \{x_j \in T_{SEdg}(\{l_1^r\}) : ||l_2^r - x_j|| \leq ||l_1 - l_2||\}$,

$Com_{l_1}^{(l_2)} = \{x \in \mathcal{D} : \tau/2 < ||x - l_1||, ||x - l_2|| \leq \tau\}$;

$ld_{SEdg}^{(l_2)} = \sum_{i=1} d_i$, $sd_{SEdg}^{(l_2)} = \sum_{i=1} d_i^2$, $d_i = ||l_1 - x_i|| > \tau/2$ and $x_i \in l_1^{(l_2)}$.

The set of shared patterns $Com_{l_1}^{(l_2)}$ between l_1^r and l_2^r is actually grouped with leader $l_1 \in \mathcal{L}$ as l_1 is observed prior to l_2. A leader l with statistics of its followers w.r.t. all other leaders is termed as *rough bubble*, $l(l^r)$ being the representative pattern. Let $\mathbb{S}$ be the set of *rough bubble*s, which is the summary of whole dataset. It is noted that followers of leader $l \in \mathcal{L}$ are also the members of a *rough bubble* $S \in \mathbb{S}$, whose representative is l^r. The whole process of summarization is depicted in Algorithm 5.2.

Complexity of the Rough bubble

The complexity of proposed summarization scheme is analyzed as follows.

i) The set $\mathcal{L}$ can be obtained in $O(mn)$ time, where $m = |\mathcal{S}|$.

ii) Computing statistics of each rough leaders (*rough bubble*) takes times of $O(n)$. For all leaders, scheme takes time of $O(mn)$. The space complexity is $O(m^2)$.

Overall time and space requirements of *rough bubble* are $O(mn)$ and $O(m^2)$, respectively.

The summary information computed in this subsection for each *rough bubble* is used in next for calculation of distances between a pair of *rough bubbles*.

5.3.3 Clustering method

The summary information obtained for a dataset in previous section is used here for clustering. Single-link method is applied to this summary information to obtain the clustering of a dataset. The single-link with *rough bubble* is termed as Rough-single-link (RSL) clustering method. Using the information of each *rough bubble*, distance between a pair of *rough bubble*s is computed. The distance function *dist* is defined as follows.

$dist : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is set of non negative real numbers.

Let the average and standard deviation of distances from l_1 to its followers w.r.t. l_2 be $\mu_{l_1}^{(l_2)}$ and $\sigma_{l_1}^{(l_2)}$, respectively. Similarly, $\mu_{l_2}^{(l_1)}$ and $\sigma_{l_2}^{(l_1)}$ be the average and standard deviation of distances from l_2 to its followers w.r.t. l_1, respectively. The $\mu_{l_1}^{(l_2)}$ and $\sigma_{l_1}^{(l_2)}$ are calculated using information available in *rough bubble* S_1 as follow.

$$\mu_{l_1}^{(l_2)} = \frac{ld^{(l_2)}}{k^{(l_2)}}; \quad \sigma_{l_1}^{(l_2)} = \sqrt{\frac{sd^{(l_2)}}{k^{(l_2)}} - (\mu_{l_1}^{(l_2)})^2}$$

Similarly, we can calculate $\mu_{l_2}^{(l_1)}$ and $\sigma_{l_2}^{(l_1)}$ for the *rough bubble* S_2.

The specific distance between a pair of *rough bubbles* S_1, S_2 (S_1 appears prior to S_2) are noted next.

I. (No "gap") If two bubbles are adjacent ($||l_1 - l_2|| < 2\tau$) to each other and following conditions hold simultaneously (5.3),

   i) Tolerance class of l_1^r intersects with tolerant shared edge of l_2^r and vice-versa.

   ii) There exists a pattern in tolerant shared edge of l_2^r, which is not a member of of tolerance class of l^r, *i.e.* ($T_{SEdg}(\{l_2^r\}) \setminus T(l_1^r)$) $\neq \emptyset$.

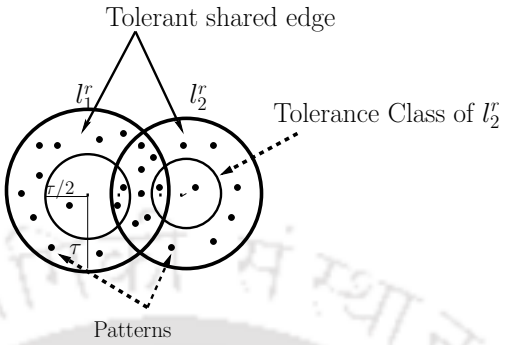


Figure 5.3: No "gap" between l_1 and l_2 rough bubbles

Then, $dist(S_1, S_2) = 0$.

II. ("Gaps") If two rough bubbles are adjacent ($||l_1 - l_2|| < 2\tau$) to each other, then one *rough bubble* (say S_1) may contain patterns from more than one clusters (Figure 5.1). As a result, S_1 may have gap (cluster separation). These gaps are handled by reassigning correct position of patterns from one *rough bubble* to other. Few scenarios can be observed as follow.

A. S_1 **contains gaps in the direction of** S_2 **:** Gaps may appear in S_1 for two types of orientations of patterns in rough bubbles (Figure 5.4).

a) If assignments of patterns in bubbles follow conditions stated below simultaneously,

i) Tolerance class of l_1^r does not intersect with tolerant shared edge of l_2^r.

ii) Tolerance class of l_2^r intersect with tolerant shared edge of l_1^r.

iii) There exist only shared patterns $Com_{l_1}^{l_2}$ in the tolerant shared edge of l_1^r. *i.e.* $(T_{SEdg}(\{l_1^r\}) \setminus Com_{l_1}^{l_2}) = \emptyset$.

In this case, the region of l_2^r is having more patterns towards l_1^r than the patterns of l_1^r w.r.t. l_2^r. Therefore, the shared patterns ($Com_{l_1}^{l_2}$) should be with S_2. The method removes the shared patterns from S_1 and patterns are added to S_2. The statistics of both *rough bubbles* (S_1, S_2) are updated accordingly.

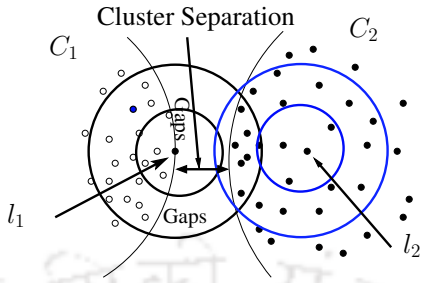


Figure 5.4: l_1 contains "gap" in the direction of l_2

$$ld^{(l_2)} = ld^{(l_2)} - |Com_{l_1}^{(l_2)}| * (\frac{ld^{(l_2)}}{k^{l_2}});$$
$$sd^{(l_2)} = sd^{(l_2)} - |Com_{l_1}^{(l_2)}| * (\frac{ld^{(l_2)}}{k^{(l_2)}})^2;$$
$$ld^{(l_1)} = ld^{(l_1)} + |Com_{l_1}^{(l_2)}| * (||l_1 - l_2|| - \frac{ld^{(l_2)}}{k^{(l_2)}});$$
$$sd^{(l_1)} = sd^{(l_1)} + |Com_{l_2}^{(l_1)}| * (||l_1 - l_2|| - \frac{ld^{(l_2)}}{k^{(l_2)}})^2.$$

b) Gaps may appear in S_1, if tolerant shared edges of l_1 and l_2 intersect each other and number of patterns in the shared edge of l_2 are more compared to the patterns in the shared edge of l_1 (Figure 5.5).
In this case re-assignment is necessary. The proposed method removes shared patterns from S_1 and adds them to S_2. Accordingly, it updates the corresponding statistics.

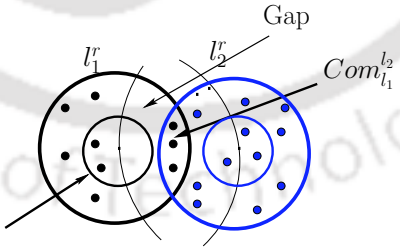


Figure 5.5: l_1 contains "gap" in the direction of l_2 (II.A.(b))

B. S_2 **contains gaps in the direction of** S_1**:** If converse of the above conditions stated in $II.$ (A) hold simultaneously, then gaps appears in S_2. However, re-assignment is not required as shared pattern $(Com_{l_1}^{l_2})$ are already grouped with S_1.

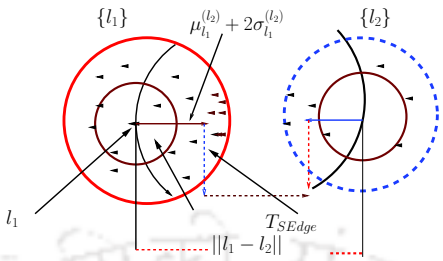


Figure 5.6: $dist(S_1, S_2) = \max(||l_1 - l_2|| - (\mu_{l_1}^{(l_2)} + 2\sigma_{l_1}^{(l_2)}) - (\mu_{l_2}^{(l_1)} + 2\sigma_{l_2}^{(l_1)}), 0)$.

According to the cases mentioned above, we restructure the *rough bubble* and recalculated $\mu_{l_1}^{(l_2)}$, $\sigma_{l_1}^{(l_2)}$, $\mu_{l_2}^{(l_1)}$ and $\sigma_{l_2}^{(l_1)}$. Distance between a pair of *rough bubble*s is defined in Definition 5.5.

Definition 5.5 (Distance between two *rough bubbles*) *The distance between a pair of rough bubbles S_1 and S_2 is defined as*
$dist(S_1, S_2) = \max\{ [\ ||l_1 - l_2|| - (\mu_{l_1}^{(l_2)} + 2\sigma_{l_1}^{(l_2)}) - (\mu_{l_2}^{(l_1)} + 2\sigma_{l_2}^{(l_1)})\ ],\ 0\ \}$.

The $dist(.)$ function satisfies following properties.

1. $dist(S_1, S_2) \geq 0$
2. $dist(S_1, S_2) = 0$, if both of them are part of same cluster or $S_1 = S_2$.
3. $dist(S_1, S_2) = dist(S_2, S_1)$. □

Agglomerative hierarchical clustering method (single-link) is applied to set of rough bubbles $\mathbb{S}$. using the distance measure defined in (Definition 5.5). This gives a hierarchy of clusterings of *rough bubble*s. Next, each *bubble* is replaced by its members. Finally, a hierarchy of clusterings of the dataset $\mathcal{D}$ is obtained. The whole process of clustering is noted in Algorithm 5.3.

Complexity Analysis

The complexity of *RSL* method is analyzed as follows.

1. The summarization scheme takes time of $O(mn)$. The space complexity is $O(m^2)$.

Algorithm 5.3 Rough-single-link$(\mathcal{D}, \tau)$

Apply *rough bubble* $(\mathcal{D}, \tau)$ as given in Algorithm 5.2.
Let $\mathbb{S} = \{S_1, S_2, \ldots, S_m\}$ be the output of *rough-bubble* $(\mathcal{D}, \tau)$.
for each leader l_i^r in $S_i \in \mathbb{S}$ **do**
  for each leader l_j^r in $S_j \in \mathbb{S} : l_i \neq l_j$ **do**
    if $(||l_i^r - l_j^r|| \leq 2\tau)$ **then**
      if (tolerance class of $l_i^r(l_j^r)$ intersects with $l_j^r(l_i^r)$ and intersection of shared edges $\neq \emptyset$) **then**
        $dist(S_i, S_j) = 0$; {Two data spheres are part of a cluster.}
      else if (tolerance class of l_i^r intersects with l_j^r and $(l_i^r \cap Com_{l_i}^{l_j}) \neq \emptyset$) **then**
        $dist(S_i, S_j) = 0$; {No reassignment of patterns}
      else if (tolerance class of l_i^r not intersect with l_j^r but tolerance class of l_j^r does and $(l_j^r \cap Com_{l_i}^{l_j}) \neq \emptyset$) **then**
        Add $Com_{l_i}^{(l_j)}$ to S_j; Remove these from S_i.
        Update $ld^{l_j}, sd^{l_j}, ld^{l_i}, sd^{l_i}, k^{(l_i)}, k^{(l_j)}$; {There is a gap in S_1 w.r.t. l_2}
      end if
      if $(Com_{l_i}^{(l_j)} \neq \emptyset$ and $(\frac{|T_{SEdg}^{(l_j)}(\{l_i^r\}) \backslash Com_{l_i}^{(l_j)}|}{|T_{SEdg}^{(l_i)}(\{l_j^r\}) \backslash Com_{l_i}^{(l_j)}|}) <= 0.5$) **then**
        Add $Com_{l_i}^{(l_j)}$ to S_j and remove from S_i;
        Updates $ld^{l_j}, sd^{l_j}, ld^{l_i}, sd^{l_i}, k^{(l_i)}, k^{(l_j)}$
      end if
    end if
    Calculate $\mu_{l_i}^{(l_j)}, \sigma_{l_i}^{(l_j)}, \mu_{l_j}^{(l_i)}, \sigma_{l_j}^{(l_i)}$
    $dist(S_i, S_j) = \max(||l_i - l_j|| - (\mu_{l_i}^{(l_j)} + 2\sigma_{l_i}^{(l_j)}) - (\mu_{l_j}^{(l_i)} + 2\sigma_{l_j}^{(l_i)}), 0)$
  end for
end for
Apply single-link to $\mathbb{S}$. Let the result be $\pi_{\mathbb{S}} = \{\pi_1, \pi_2, \ldots, \pi_{|\mathbb{S}|}\}$, set of clusterings of $\mathbb{S}$.
Expand each $S_i \in \pi_i \in \pi_{\mathbb{S}}$. Therefore, $\pi_{\mathcal{D}=\{\pi_{D_1}, \pi_{D_2}, \ldots, \pi_{D_{|\mathbb{S}|}}\}}$.
Output $\pi_{\mathcal{D}}$

2. Calculating distance-matrix for *rough bubble* takes $O(m^2)$ time.

3. Clustering the bubbles using single-link method takes $O(m^2)$.

Overall time and space complexity of *RSL* are $O(mn)$ and $O(m^2)$, respectively. The proposed method scans the dataset twice.

Table 5.1: Dataset used to show effectiveness of rough bubble scheme

Data Set	# Patterns	# Features
Spiral(Synthetic)	3330	2
Pendigits	7494	16
Shuttle	58000	9
Circle(Synthetic)	28000	2
GDS10	23709	28

5.4 Experimental Results

This section presents performance analysis of proposed computation reduction technique and data summarization scheme. Experiments are conducted with synthetic as well as real world dataset (`http://archive.ics.uci.edu/ml/`, (`http://www.ncbi.nlm.nih.gov/geo/`)) without class labels. Brief description of the dataset are given in Table 5.1. Spiral and Circle are 2-dimensional synthetic dataset discussed in previous chapters. As discussed in section 5.3.2 TI-LEADER is used in finding representative points of *rough bubble*. Therefore, experimental results of TI-LEADER are presented separately. First, TI-LEADER is evaluated and followed by *rough bubble*.

5.4.1 Performance of TI-LEADER method

The TI-LEADER and classical leaders (without distance matrix) clustering methods are executed on Intel Core 2 Duo CPU (3.0 GHZ) with 2GB RAM IBM PC. The value of τ is determined based on intended compression ratio of the proposed summarization scheme.

Experiments with Spiral dataset (synthetic) are reported in Table 5.2. With leaders threshold $\tau = 10$, the *TI-LEADER* calculates 2.33 million less distance computations compared to the classical leaders method. However, both methods achieve same set of leaders. It is found that proposed distance reduction technique performs one order of magnitude less distance computations than the leaders method (Table 5.2).

To show performance analysis of *TI-LEADER* with real dataset, it is tested with Shuttle dataset. With leaders threshold $\tau = 0.0005$, *TI-LEADER* makes 355 millions

Table 5.2: Performance of TI-LEADER for Spiral Dataset

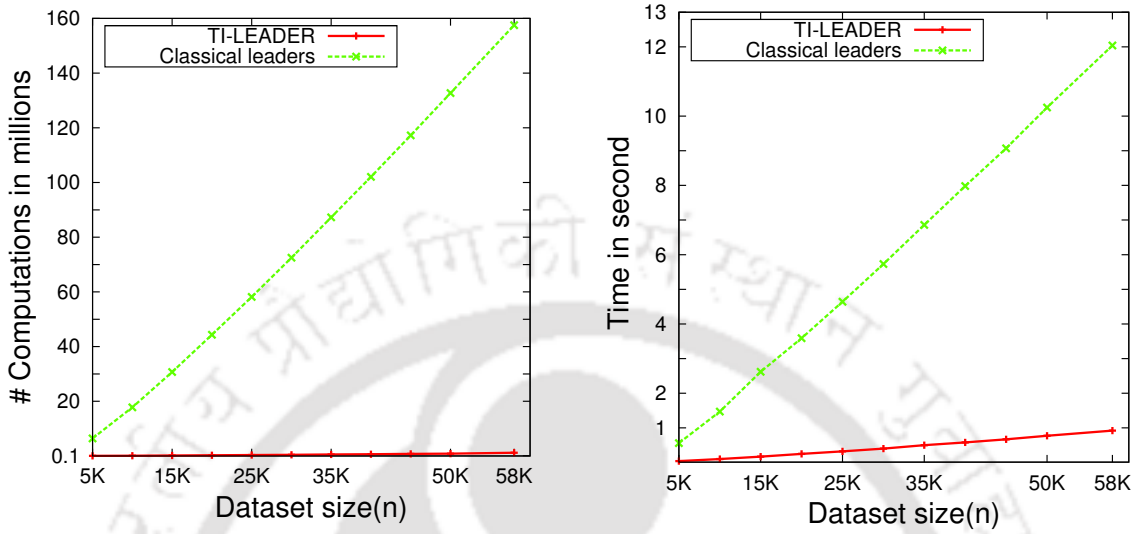
Threshold(τ)	Method	# Computations (in Million)	Speed-up (in Computations)
10	leaders	2.42	26.60
	TI-LEADER	**0.09**	
15	leaders	1.15	17.70
	TI-LEADER	**0.06**	
20	leaders	0.80	12.90
	TI-LEADER	**0.06**	
25	leaders	0.55	10.78
	TI-LEADER	**0.05**	

less distance calculations compared to the classical leaders method. The *TI-LEADER* is more than one order of magnitude faster than the classical leaders method with $\tau = 0.0005$ (Table 5.3). With other values of τ (0.001, 0.002), it is found that proposed *TI-LEADER* technique computes significantly less distance calculations compared to the classical approach.

Table 5.3: Performance of TI-LEADER for Real Dataset

Dataset	Threshold (τ)	Method	# Computations (in Million)	Time (in Sec.)
Shuttle	0.0005	Leaders	356.9	29.37
		TI-LEADER	**1.0**	**1.60**
	0.0010	Leaders	157.5	12.03
		TI-LEADER	**1.0**	**0.92**
	0.0020	Leaders	75.1	5.57
		TI-LEADER	**0.9**	**0.60**

To show the performance of the proposed computation reduction technique with variable dataset size, experiments are conducted with Shuttle dataset with leaders threshold $\tau = 0.001$. It may be noted that with increase of the dataset size, number of distance calculations by the *TI-LEADER* reduces significantly compared to that of the classical leaders (Fig. 5.7(a)). Figure 5.7(b) shows time taken by the leaders method and *TI-LEADER* method for Shuttle dataset. It may be noted that the execution time of the



(a) Number of distance computations Vs dataset size

(b) Execution time Vs dataset size

Figure 5.7: Experiments with Shuttle dataset at $\tau = 0.001$

TI-LEADER improves significantly compared to the classical leaders.

5.4.2 Performance of rough bubble summarization Scheme

To show effectiveness of proposed *rough bubble*, clustering methods namely, classical single-link, single-link with directional data bubble ($\overrightarrow{dbSL}$) *RSL*, SSL (proposed in Chapter 4) methods are implemented.

Table 5.4 shows the detailed performance of the four clustering methods for different dataset. Compression Ratio (CR) for *rough bubble* is defined as $CR = n/m$, where n and m are the cardinality of dataset and a representative set of the dataset. Rand Index (RI) and Purity are computed to compare clusterings produced by these methods. For this purpose, a clustering is considered as the final output in which number of clusters are same as the number of classes of a given dataset for each method. RI and $Purity$ (Table 5.4) are computed between the clusterings (partitions) produced by single-link and *RSL*, SSL, $\overrightarrow{dbSL}$ method separately.

With Spiral dataset, RSL and SSL methods produce same clusterings as produced

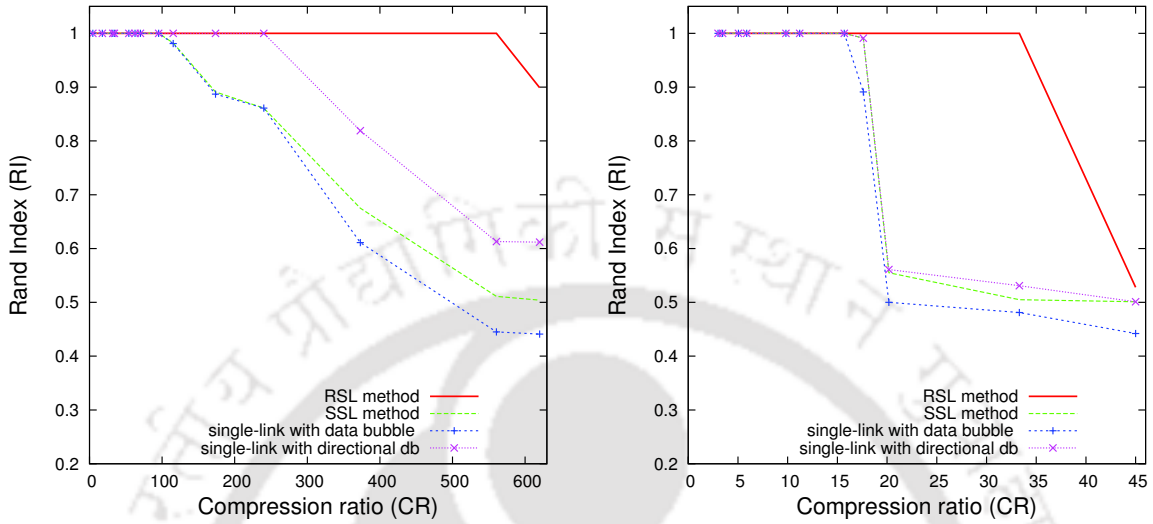
Table 5.4: Results for synthetic dataset

Dataset	*Compression Ratio*($\frac{n}{m}$)	Method	Time (in sec.)	Rand Index (RI)	Purity
Spiral	17.61	RSL	0.06	1.000	1.000
	17.61	SSL	0.01	1.000	1.000
	17.61	$\overrightarrow{db}SL$	0.05	0.996	0.999
	–	single-link	22.12	–	–
Circle	373.5	RSL	0.19	1.000	1.000
	373.5	SSL	0.05	0.670	0.681
	373.5	$\overrightarrow{db}SL$	0.14	0.819	0.919
	–	single-link	975.12	–	–

by the single-link method ($RI = 1.000, Purity = 1.000$) with compression ratio 17.61 (Table 5.4). With high compression ratio ($CR = 373.5$), RSL clustering method produces same clusterings as produced by that of the single-link method with Circle dataset (Table5.4). However, RSL method is more than three order of magnitudes faster than that of the classical single-link method. The SSL method is more than four order of magnitudes faster than classical single-link at same compression ratio (373.5). However, clustering results of SSL ($RI = 0.670$) are affected significantly. The RSL method outperforms SSL and $\overrightarrow{db}SL$ method in clustering results for both dataset namely, Spiral and Circle in expense of little execution time.

Rand Index computed from clustering results produced by RSL, SSL, single-link with data bubble and single-link with directional bubbles are shown in Figure 5.8 as compression ratio varies for Circle and Spiral dataset. It may be noted that proposed RSL can produce consistently good clustering results ($RI = 1.000$) with high compression ratio.

The clustering similarity w.r.t. classical single-link at different levels of hierarchy (dendogram) produced by RSL are reported in Table 5.5 for Circle dataset. It is found that RSL is superior than $\overrightarrow{db}SL$ and SSL methods.



(a) Circle dataset (b) Spiral dataset

Figure 5.8: Rand Index (RI) Vs Compression Ratio (CR)

Table 5.5: Clustering hierarchy obtained using three summarization schemes

Number of cluster	Method	Rand Index (RI)	Purity
2	RSL	1.000	1.000
	SSL	1.000	1.000
	$\overrightarrow{db}SL$	1.000	1.000
3	RSL	1.000	1.000
	SSL	0.627	0.716
	$\overrightarrow{db}SL$	0.744	0.750
4	RSL	1.000	1.000
	SSL	0.852	0.913
	$\overrightarrow{db}SL$	1.000	1.000
5	RSL	0.971	0.981
	SSL	0.835	0.963
	$\overrightarrow{db}SL$	0.949	0.925
6	RSL	0.962	0.981
	SSL	0.827	0.897
	$\overrightarrow{db}SL$	0.921	0.876
7	RSL	0.931	0.980
	SSL	0.813	0.860
	$\overrightarrow{db}SL$	0.892	0.839

Table 5.6: Results for real world dataset

Dataset	*Compression Ratio*($\frac{n}{m}$)	Method	Time (in sec.)	Rand Index (RI)	Purity
Pendigits	5.50	RSL	5.20	0.985	0.999
	5.50	SSL	1.07	0.984	0.999
	5.50	$\overrightarrow{dbSL}$	4.57	0.985	0.998
	–	single-link	512.78	–	–
	8.50	RSL	2.43	0.985	0.999
	8.50	SSL	0.31	0.984	0.999
	8.50	$\overrightarrow{dbSL}$	1.80	0.984	0.999
	–	single-link	512.78	–	–
	17.07	RSL	0.80	0.985	0.999
	17.07	SSL	0.11	0.983	0.999
	17.07	$\overrightarrow{dbSL}$	0.66	0.979	0.990
	–	single-link	512.78	–	–
GDS10	10.64	RSL	31.6	0.930	0.992
	10.64	SSL	7.68	0.818	0.917
	10.64	$\overrightarrow{dbSL}$	26.33	0.887	0.961
		single-link	4215.51	-	-
	19.23	RSL	13.55	0.901	0.981
	19.23	SSL	1.88	0.801	0.892
	19.23	$\overrightarrow{dbSL}$	9.88	0.881	0.955
		single-link	4215.51	-	-
Shuttle (40,000)	18.70	RSL	58.70	0.919	0.997
	18.70	SSL	30.12	0.740	0.749
	18.70	$\overrightarrow{dbSL}$	53.40	0.819	0.918
		single-link	7101.12	-	-

Experiments with real world large dataset

Performance of *rough bubble* are evaluated with three real world dataset (Pendigits, Shuttle and GDS10). Pendigits and Shuttle are taken from UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/`). We took $40,000$ patterns out of $58,000$ of Shuttle dataset for our experiments due to memory shortage. To store the entire distance matrix for whole dataset, single-link method consumes $58000 \times 58000 \times 4 = 12.54GB$ memory.

From GEO database (`http://www.ncbi.nlm.nih.gov/geo/`) a dataset named GDS10 is selected having removed 'null' values. Finally, $23,709$ out of $39,114$ patterns are used for experiments. The experimental results are reported in Table 5.6 with different CR

values.

In Pendigits dataset, RSL method is up to two order of magnitudes faster than classical single-link method and clustering results are very close ($RI = 0.985$, $Purity = 0.999$ at $CR = 5.5$) to that of the single-link method (Table 5.6). The $\overrightarrow{db}SL$ produces same clustering as the results produced by RSL method and execution time of $\overrightarrow{db}SL$ is marginally less than RSL method. Clustering results produced by SSL method is close to RSL and $\overrightarrow{db}SL$ methods. The SSL takes less computation time compared to RSL and $\overrightarrow{db}SL$ methods.

Experiments with Shuttle dataset shows that RSL method is significantly faster (more than 100 times) than classical single-link method (Table 5.6). Execution time of RSL is at par with the $\overrightarrow{db}SL$ method. Experimental results follow similar trends with GDS10 dataset (Table 5.6).

5.5 Conclusions

Data summarization is an effective way of speeding up clustering methods for large dataset. The *rough bubble* summarization scheme is introduced to speed up single-link method. The summarization scheme exploits leaders method and tolerance rough set theory. The proposed *rough bubble* scans the dataset twice only. Clustering results produced by RSL method are very close to the results produced by classical single-link method and it takes significantly less time compared to that of the single-link method. The RSL outperforms directional data bubble based single-link method at clustering results while trading little computation time.

Proposed *data sphere* and *rough bubble* summarization schemes successfully speed up distance based single-link method in large dataset. However, these summarization schemes combined with single-link cannot identify clusters with different distributions and in presence of outliers in a dataset. Therefore, a contribution is made in the next chapter to discover clusters with largely varied densities in a dataset.

Chapter 6

A density based clustering method for arbitrary shaped clusters

6.1 Introduction

Many clustering methods have been developed over the years in various application domains. These clustering methods are categorized from different perspectives. An algorithmic perspective categories clustering methods based on the types of algorithms (optimization criteria) used to produce clustering outputs- distance based method (global criteria) and (local criteria) density based method.

A Distance based clustering method optimizes a global criteria, which is the function of distance between patterns in a dataset. Most of the distance based methods cannot find arbitrary shaped clusters in a dataset.

Density based clustering methods search dense regions in feature space using local density information. A dense region separated by less dense (sparse) region is considered as a cluster in this approach. All such dense regions are treated as clusters of a dataset. Density based clustering methods find density information (characteristics) of each data point of a given dataset. Based on the characteristics, clusters or outlier points[1] are

[1] A point is said to be an outlier if it does not conform well defined notion of normal behavior [Chandola et al. (2009)]

obtained by this approach. Main advantage of this approach is that it can find clusters of arbitrary shapes and sizes in a dataset. Density based clustering methods is used in many applications such as geographic information systems, medical imaging, and weather forecasting.

DBSCAN (A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise) [Ester et al. (1996)] is a classical density based clustering method, which can detect clusters of arbitrary shapes and sizes along with outlier points. DBSCAN is very sensitive to input parameters- ϵ (radius of neighborhood of a point), $Minpts$ (minimum number of data points in the neighborhood). DBSCAN works well in a dataset with clusters situated away from each other. However, it performs poorly if clusters are adjacent to each other in the feature space. It cannot produce effective outputs (proper clustering) when variability of densities of clusters is high in a dataset.

A neighborhood based clustering (NBC) method is introduced to overcome the drawbacks of DBSCAN in [Zhou et al. (2005)]. The NBC needs only one input parameter K, i.e. number of nearest neighbors. It can discover clusters of arbitrary shape and sizes in a dataset with variable density. The core idea of NBC is Neighborhood Density Factor (NDF) which measures local density of a point. NDF of a point is ratio of the number of reverse K nearest neighbors to the number of K nearest neighbors of the point. However, NDF does not account relative position (distance) of a point in its neighborhood. As a result, NBC may not detect border points correctly and the method is sensitive to K values. M.Kryszkiewicz et al. showed that this method could not work satisfactory with high dimensional large dataset [Kryszkiewicz and Lasek (2010a)].

In this chapter, a density based clustering method referred to as *Nearest Neighbor Factor based Clustering (NNFC)* is proposed. NNFC can find clusters of arbitrary shaped and sizes in variable density dataset. NNFC method is developed based on a factor termed as *Nearest Neighbor Factor (NNF)*. NNF determines relative 'importance' of an object in its K neighborhood using relative distance and density information of the neighborhood. Clustering results produced by the proposed NNFC clustering method is very close to the natural clusters of the dataset (as per Purity measure). It is less sensitive to parameter

K and suitable for large dataset. The contributions of this chapter are summarized as follow.

- A novel parameter termed as *Nearest Neighbor Factor (NNF)* is introduced to determine relative 'importance' of an object in its neighborhood region. Unlike NDF, NNF accounts closeness (distance between a point and its neighbor) as well as neighborhood density of the neighbor.
- Proposed NNFC clustering method can find clusters in variable density dataset and less sensitive to input parameters K.
- NNFC can find outlier points in a dataset.
- *NNFC* is tested with synthetic and real world dataset and it is found that *NNFC* outperforms TI-K-Neighborhood-Index supported NBC [Zhou et al. (2005)] in clustering quality as well as execution time.
- Clustering results produced by *NNFC* is close to natural clustering of the dataset (as per Purity measure)

Rest of the chapter is organized as follows. Section 6.2 describes a summary of related work. Section 6.3 describes proposed clustering method, with experimental results and discussion in section 6.4. Finally, section 6.5 concludes the paper.

6.2 Related work

This section presents a brief summary of popular density based clustering methods (discussed in Section 2.3.5). A detailed discussion is presented here also on clustering methods closely related to the method proposed in this chapter.

- **Density Based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al. (1996)]:** DBSCAN is a widely used density based clustering method for finding clusters of arbitrary shapes and sizes. Main idea of DBSCAN is *denseness* and *connectivity*, which are computed using local distribution. A point $x \in \mathcal{D}$ is said

to be dense in the feature space if a hyper-sphere of radius ϵ ($\in \mathbb{R}^+$) centered at x has at least $Minpts$ ($\in \mathbb{N}$) numbers of points. DBSCAN follows certain *connectivity relation*s between points, which are listed below.

i) *density reachable:* A point y is density reachable from x, if there is a sequence of patterns $x_1 = x, x_2, \ldots, x_{m>1} = y$ such that $||x_i - x_{i+1}|| \leq \epsilon$ and each x_i, $i = 1..(m-1)$ is dense. It may be noted that point y can also be dense one.

ii) *density connected:* A pair of points x and y are said to be density connected if there exists a dense point x_c such that both x and y are density reachable from point x_c.

A cluster is defined as a set of density connected points which is maximal w.r.t. density reachibility. Finally, non-dense points, which are not included to any cluster are declared as outlier points. However, it has following drawback:

i) Determining values of input parameters ϵ, $Minpts$ are not easy task for real world large dataset. Moreover, different parts of dataset could require different parameters set. Clustering results are sensitive to the parameters.

ii) It fails to detect clusters of arbitrary shapes in a dataset when clusters of different densities exist.

iii) DBSCAN relies on the output of R*-tree for range queries. However, R*-tree is suitable for low dimensional dataset.

- **Ordering Points To Identify the Clustering Structure (OPTICS) [Ankerst et al. (1999)] :** To overcome some of the drawbacks of the DBSCAN method, OPTICS was introduced in [Ankerst et al. (1999)]. It creates an augmented *ordering* of points in dataset, which represents hierarchical clustering structures of the dataset. This method is seen as multiple runs of DBSCAN with different $\epsilon' \leq \epsilon$ and $Minpts$. However, finding ϵ and $Minpts$ are difficult task. It relies on R*-tree for range queries.

- **DENsity-based CLUstEring (DenClue) [Hinneburg and Keim (1998)]:** DenClue works with density function is the superposition of several *influence functions*. The influence function describes an importance of a dataset point within its neighborhood region. It generalizes a set of clustering methods by defining various influence functions. A Clusters is a sequence of dataset points with local (maxima) density more than pre-specified threshold. However, DenClue requires many parameters which need to be adjusted for proper clustering output.

- **Neighborhood-Based Clustering (NBC)[Zhou et al. (2005)]:** NBC is a density based clustering method, which needs only one input parameter K *i.e.* number of nearest neighbors of a point. NBC uses only neighborhood information to find clusters of arbitrary shapes, size with variable densities. The core idea of NBC is *Neighborhood Based Density Factor (NDF)*, which measures local density of a dataset point. NDF of a point p is a function of reverse K-nearest neighbors and of K-nearest neighbors (equation (6.1)). Reverse K-nearest neighbors of a point p is the set of patterns whose K-nearest neighbors contain p.

$$NDF(p) = \frac{\text{Cardinality of reverse } K\text{-nearest neighbors of p}}{\text{Cardinality of } K\text{-nearest neighbors of p}} \tag{6.1}$$

  Based on the value of NDF, a point can be classified into any one of the three categories: *dense point* ($NDF(p) > 1$), *sparse point* ($NDF(p) < 1$) and *even point* ($NDF(p) = 1$). Based on value of NDF, NBC defines relation of *density reachable* and *density connected* in the similar line of DBSCAN as follow.

  i) *density reachable:* A point y is density reachable from x, if there is a sequence of dense or even points $x_1 = x, x_2, \ldots, x_{m>1} = y$ such that x_{i+1} belongs to K nearest neighbor of x_i, $i = 1..(m-1)$.

  ii) *density connected:* A pair of points x and y are said to be density connected if there exists a dense point x_c such that both x and y are density reachable from point x_c.

It finds clusters by checking NDF value of each point in a dataset. If a point p is dense or even point, a new cluster with p is created. NBC then recursively adds each K -neighbor point to this cluster by checking NDF values. This process stops when there is no dense or even points. Method can find outliers in the dataset. However, NBC uses only K-neighbors and reverse k-neighbors counts. It does not use relative distance (position) of a point in its neighborhood region. Therefore, with a slight change in value of K, it produces different clustering results and it treats genuine clusters as outlier points (experimentally demonstrated in Section 6.4). Kryszkiewicz and Lasek [Kryszkiewicz and Lasek (2010a)] pointed out that NBC suffers computation burden with high dimensional dataset.

- **TI-K-Neighborhood-Index based NBC [Kryszkiewicz and Lasek (2010a)]:** Kryszkiewicz and Lasek [Kryszkiewicz and Lasek (2010a)] introduced an spatial indexing scheme in metric space to speed up NBC method in large high dimension dataset. The TI-K-Neighborhood-Index based NBC has two sequential steps. In the first step, it finds K nearest neighbors of each point using TI-K-Neighborhood-Index, which is discussed next.

  TI-K-Neighborhood-Index [Kryszkiewicz and Lasek (2010a)] is an useful technique to quickly find K nearest neighbors of a point in a dataset. It is based on the property of triangle inequality of metric space. Naive approach for finding K nearest neighbors of a point p needs to compute $n-1$ distance calculations. Therefore, time complexity of naive for all points in a dataset is $O(n^2 * N)$, where N is the dimension of the dataset of size n. However, TI-K-Neighborhood-Index quickly picks potential K neighbors points by discarding a large number of points for finding K nearest neighbors of a point. It is independent of dimension. The indexing technique works as follows.

  Initially, dataset points are arranged in a list L in sorted order (ascending) with respect to their distances from origin (norm). For finding K nearest neighbors of a point $x \in \mathcal{D}$, indexing technique first identifies position of x in the list. Subsequently, it finds K points such that difference of norms of x and those points ($||x|| - ||q||$,

$q \in L$) are minimum among the points in L. These K points determine a radius EPS of a sphere centered at x, which covers all K nearest neighbors of x. The value of EPS is the maximum of actual distances between x and those K points.

The TI-K-Neighborhood-Index collects potential nearest neighbor set of x applying Theorem 6.1. In this process, it adds a preceding or succeeding point q to potential nearest neighbor set such that norm difference is not more than EPS ($||x|| - ||q|| \leq EPS$). Finally, actual distances are computed from x to each point in potential nearest neighbors set and K minimum distance points are declared as K nearest neighbors of x. The whole approach is depicted in Algorithm 6.1.

Theorem 6.1 (Kryszkiewicz and Lasek (2010a)) *Let L be a set of points ordered in a non-decreasing way with respect to their magnitudes ($||.||$) of the points. Let p be any point in L, and $EPS \in \mathcal{R}^+$ be a value such that $|N_{EPS}(p)^2| >= K$. Let q_f and q_b be preceding and succeeding points of p in the ordered set L, respectively such that $||q_f|| - ||p|| > EPS$ and $||p|| - ||q_b|| > EPS$.Then*

1. *q_f and all points following q_f do not belong to K nearest neighbors of p.*
2. *q_b and all points preceding q_b do not belong to K nearest neighbors of p.* □

They showed that this indexing based NBC method outperforms NBC supported VA file [Weber et al. (1998)] and NBC based R-tree [Guttman (1984)].

6.3 Proposed clustering method

In this section, a formal definition for *Nearest Neighbor Factor (NNF)* is given. Subsequently, proposed density based method $NNFC$ is discussed in details.

Proposed $NNFC$ uses K nearest neighbors statistics of each point to determine its position in its neighborhood region. For the shake of readability, definition of K nearest neighbors is provided and followed by average K nearest neighbor distance. Using these

[2] $N_{EPS}(p) = \{q \in D \mid q \neq p,\ ||p - q|| \leq EPS\}$

Algorithm 6.1 *TI-K-Neighborhood-Index*($\mathcal{D}$, K)

for each pattern $x \in \mathcal{D}$ **do**
  Calculate $||x||$
end for
Make a list L of all sorted patterns, which are ordered in ascending order with respect to $|\ |\ .\ |\ |$
for each pattern $x \in L$ **do**
  Pick K patterns q preceding and succeeding x such that $||x|| - ||q||$ is minimum.
  Calculate $EPS = \max_{i..K} ||x - q_i||$
  KNN Set $(x) = \emptyset$
  Let p_i be the closest pattern preceding x in list L
  while $(||x|| - ||p_i|| \leq EPS)$ **do**
    if $(||x - p_i|| \leq EPS)$ **then**
      KNN Set(x) = KNN Set $\cup\{p_i\}$ and store distance between x and p_i
      if (There exists an immediate preceding point p_{i-1} in L) **then**
        $p_i = p_{i-1}$
      end if
      EPS = maximum of computed distance between x and $p \in$ KNN Set(x).
    end if
  end while
  Let q_i be the closest pattern succeeding x in list L.
  while $(||x|| - ||q_i|| \leq EPS)$ **do**
    if $(||x - q_i|| \leq EPS)$ **then**
      KNN Set(x) = KNN Set $\cup\{q_i\}$ and store distance between x and q_i
      if (There is an immediate succeeding point q_{i+1} in L) **then**
        $q_i = q_{i+1}$
      end if
      EPS = maximum of computed distance between x and $q \in$ KNN Set(x).
    end if
  end while
  Sort all points in KNN Set and output first K points as K nearest neighbors of x
end for

information, *Nearest Neighbor Factor (NNF)* is introduced. *NNF* is the key concept of the proposed clustering method. Let $\mathcal{D}$ and d be a dataset and a distance function, respectively.

Definition 6.1 (*K Nearest Neighbor (KNN) Set*) **:** *Let x be a point in $\mathcal{D}$. For a natural number K, a set $NN_K(x) = \{q \in \mathcal{D} | d(x,q) \leq d(x,q^{'}), q^{'} \in \mathcal{D}\}$ is called KNN of x if the following two conditions hold.*

(i) $|NN_K(x)| > K$ if $q^{'}$ is not unique in $\mathcal{D}$ or $|NN_K| = K$, otherwise.

(ii) $|NN_K(x) \setminus N^{(q^{'})}| = K - 1$, where $N^{(q^{'})}$ is the set of all $q^{'}$ point(s). □

The *KNN Set*$(x) \subset \mathcal{D}$ is a set of K nearest points from x in $\mathcal{D}$. *KNN Set*(x) includes all K^{th} nearest points if K^{th} nearest point is not unique.The *KNN Set* of x is also called

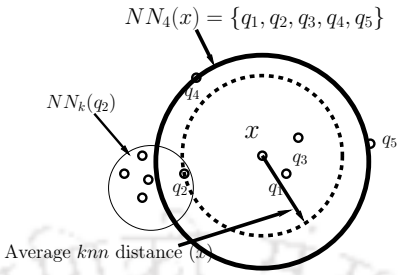


Figure 6.1: The K nearest neighbor of x with $K = 4$

neighborhood of x. In this chapter, we use it interchangeably.

Definition 6.2 (*Average KNN distance*) **:** *Let $NN_K(x)$ be the KNN set of a point $x \in \mathcal{D}$. Average KNN distance of x is average of distances between x and $q \in NN_K(x)$, i.e.* $Average\ KNN\ distance\ (x) = \frac{\sum_q d(x, q \mid q \in NN_K(x))}{|NN_K(x)|}$ □

Average KNN distance of a point x is average of distances between x and its K neighbors (Figure 6.1). If *Average KNN distance* of x is less compared to other point y, it indicates that x's neighborhood region is more densed compared to the region where y resides.

Using information of K nearest neighbors and *Average KNN distance*, we define a new parameter called *Nearest Neighbor Factor* (NNF), which is the key idea of the proposed method.

Definition 6.3 (*Nearest Neighbor Factor* (NNF)) **:** *Let x be a point in $\mathcal{D}$ and $NN_K(x)$ be the KNN of x. The NNF of x with respect to $q \in NN_K(x)$ is the ratio of $d(x, q)$ and Average KNN distance of q.*

$$NNF(x, q) = d(x, q)/Average\ KNN\ distance(q)$$

□

The NNF of x with respect to a neighbor q is the ratio of distance between x and q, and *Average KNN distance* of q. $NNF(x, q)$ indicates x's relative position with respect to q's neighborhood. Based on values of NNF, our method $NNFC$ can identify regions

with variable densities, outlier points and inlier points (cluster point). NNF values of a point $x \in \mathcal{D}$ w.r.t. to all its neighbors are close to 1 indicates that neighborhood regions of all neighbors belong to a cluster and x is an inlier point of the cluster. If NNF values of x varies significantly with respect to its neighbors, then neighborhood regions of x's neighbors may belong to different clusters. Therefore, average of NNF values of a point is considered instead of individual values. Average of NNF values of x termed $NNF_{avg}(x)$ is calculated using equation (6.2).

$$NNF_{avg}(x) = \frac{\sum_q NNF(x, q \mid q \in NN_K(x))}{|NN_K(x)|} \tag{6.2}$$

The value of $NNF_{avg}(x)$ indicates x's relative position with respect to its neighbors. It can be shown that for uniformly distributed points, value of NNF_{avg} of each point is 1. However, value of NNF_{avg} varies in real world dataset and it can be less than or more than 1 depending upon density of the region where point resides.

It is trivial to mention that $NNF_{avg}(x) = 1$ indicates that x and all its neighbors are in a cluster. Whereas, in case of $NNF_{avg}(x) \approx 1$, point x may club with a cluster of its neighbors. Otherwise, x can be noise point or another cluster point. Considering these observations, following positions of a point $x \in \mathcal{D}$ with respect to a cluster are defined. The parameter δ is used to define closeness of a point to a cluster.

Definition 6.4 (Border Point of a Cluster (BP)) : *Point x is called a border point of a cluster if $NNF_{avg}(x) < 1-\delta$ and there exists a point y such that $1-\delta \leq NNF_{avg}(y) \leq 1+\delta$ and $y \in NN_K(x)$.* □

Definition 6.5 (Core Point or Cluster Point (CP)) : *Point x is a Core Point or Cluster Point (CP) if $1-\delta \leq NNF_{avg}(x) \leq 1+\delta$.* □

Core point resides inside in a cluster, whereas border point at boundary region of a cluster. A border point may be at the boundaries of more than one clusters. However, our method considers it in a cluster whose core point discovers it first.

Definition 6.6 (Outlier Point(OP)) : *Let x be a point in $\mathcal{D}$. The point x is a Outlier (OP) if $NNF_{avg}(x) > 1+\delta$ and there is no CP in its KNN set.* □

It may be noted that definition of outlier given here is in the line with Chandola et al. [Chandola et al. (2009)]. Using the above classifications of the dataset points in the similar line of DBSCAN, following relationships between any two arbitrary points x, y in $\mathcal{D}$ are established and exploit them in the proposed method using only K neighborhood information.

Definition 6.7 (Directly Reachable) : *Let $NN_K(x)$ be KNN set of x. The point y is directly reachable from x if x is a CP and $y \in NN_K(x)$.* □

In general, directly reachable is an asymmetric relation. However, if y is a core point, then directly reachable is symmetric one.

Definition 6.8 (Reachable) : *A point y is reachable from x if there is a sequence of points $p_1 = x, p_2, \ldots, p_m = y \in \mathcal{D}$ such that p_{i+1} is directly reachable from p_i, where $i = 1..(m-1)$* □

Reachable is a transitive relation and it is a symmetric if y is a core point. Otherwise,it is asymmetric. Starting from a core point of a cluster, one can visit all points of a cluster using reachable relation. It may be noted that reachable relation is formed surrounding a core point. However, two border points of a cluster can also be related through a core point of the cluster. This relation called neighborhood connected is defined as follows.

Definition 6.9 (Neighborhood Connected) : *Two points x and y are called neighborhood connected if any one of the following conditions holds.*
(i) y is reachable from x or vice-versa.
(ii) x and y are reachable from a core point in the dataset. □

Obviously, two core points in a cluster are neighborhood connected. Now, we can define Nearest Neighbor Factor based cluster and outlier points as follow.

Definition 6.10 (*Neareat NeighborFactor based Cluster*($NNFC$)) : *Let $\mathcal{D}$ be a dataset. $C \subset \mathcal{D}$ is a NNFC cluster such that*
(i) if $x, y \in C$, x and y are neighborhood connected and (ii) if $x \in C$, $y \in \mathcal{D}$ and y is reachable from x, then y also belongs to NNFC cluster C. □

Definition 6.11 (NNFC based Outlier) : *Let $C_1, C_2, \ldots, C_m$ be NNFC clusters of dataset $\mathcal{D}$. The NNFC based outliers set is defined as $\mathcal{O} = \{p \in \mathcal{D} \mid p \notin C = \cup_{i=1}^{m} C_i\}$* □

Algorithm 6.2 *NNFC($\mathcal{D}$, K,δ)*

```
/* D is a dataset, K is the value of K nearest neighbor, δ is variation parameter */
for each pattern x ∈ D do
  Calculate NNF_avg(x) using Indexing scheme discussed in Algorithm 6.1.
end for
Each point is marked as 'undiscovered'.
cluster_id = 0;
for each pattern x ∈ D do
  if x is not marked as 'discovered' then
    Mark x as 'discovered'.
    if NNF_avg(x) < 1 − δ or NNF_avg(x) > 1 + δ then
      Mark x as 'noise'.
    else
      /*   x is a CP and start expanding a new cluster with id cluster_id + 1   */
      cluster_id = cluster_id + 1; QUEUE = ∅.
      for each p ∈ NN_K(x) do
        if p is 'undiscovered' or marked as 'noise' then
          Mark p with 'discovered' and cluster number cluster_id
        end if
        if p is a core point then
          QUEUE = QUEUE ∪ {p}.
        end if
      end for
      while QUEUE is not empty do
        Take a pattern y from QUEUE.
        for each q ∈ NN_K(y) do
          if q is 'undiscovered' or marked as 'noise' then
            Mark q with 'discovered' and cluster number cluster_id
          end if
          if q is a core point then
            QUEUE = QUEUE ∪ {q}.
          end if
        end for
        Remove y from QUEUE.
      end while
    end if
  end if
end for
```

The *NNFC* has two major steps as shown in Algorithm 6.2. In first step, it searches K nearest neighbors and computes NNF_{avg} of each pattern in the dataset. To speed up the searching process, TI-K-Neighborhood-Index [Kryszkiewicz and Lasek (2010a)] is employed for finding K nearest neighbors.

All points are marked as 'undiscovered' in the beginning of second phase. *NNFC* picks an 'undiscovered' point (say, x) and marks x as 'noise' and 'discovered', if x is not a core point (CP). Otherwise, if x is a CP, it starts expanding a new cluster by finding all reachable points from x. A reachable point which is either 'undiscovered' or 'noise' is included into the new cluster. Each point in the cluster is marked with $cluster\_id + 1$ (initialized with $cluster\_id = 0$) and 'discovered'. It may be noted that a point earlier marked as 'noise' is included as a border point of the current cluster. Similarly, a reachable point which was earlier included as a border point of other cluster is not considered for the current cluster. This process continues until all points are marked as 'discovered'. Finally, points which are marked as 'noise' are declared as outlier set and points which are marked with *cluster_id* as cluster points. Needless to mention that *NNFC* finds arbitrary shaped clusters in the dataset.

Time and space complexity of *NNFC* is analyzed as follows. Time complexity of the first step of *NNFC* is same as the complexity of TI-K-Neighborhood-Index. In the second step, *NNFC* traverses each point once and either includes it to a cluster or considers the point as a temporary outlier. Therefore, Time complexity of this step is $O(nK) = O(n)$. Overall complexity of *NNFC* is complexity of TI-K-Neighborhood-Index $+O(n)$. Space complexity of *NNFC* is space requirement of TI-K-Neighborhood-Index $+O(nN)$, where N is the dimension of dataset.

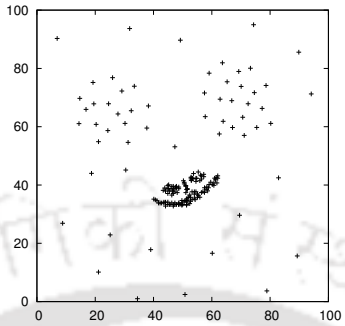


Figure 6.2: Dataset [Zhou et al. (2005)]

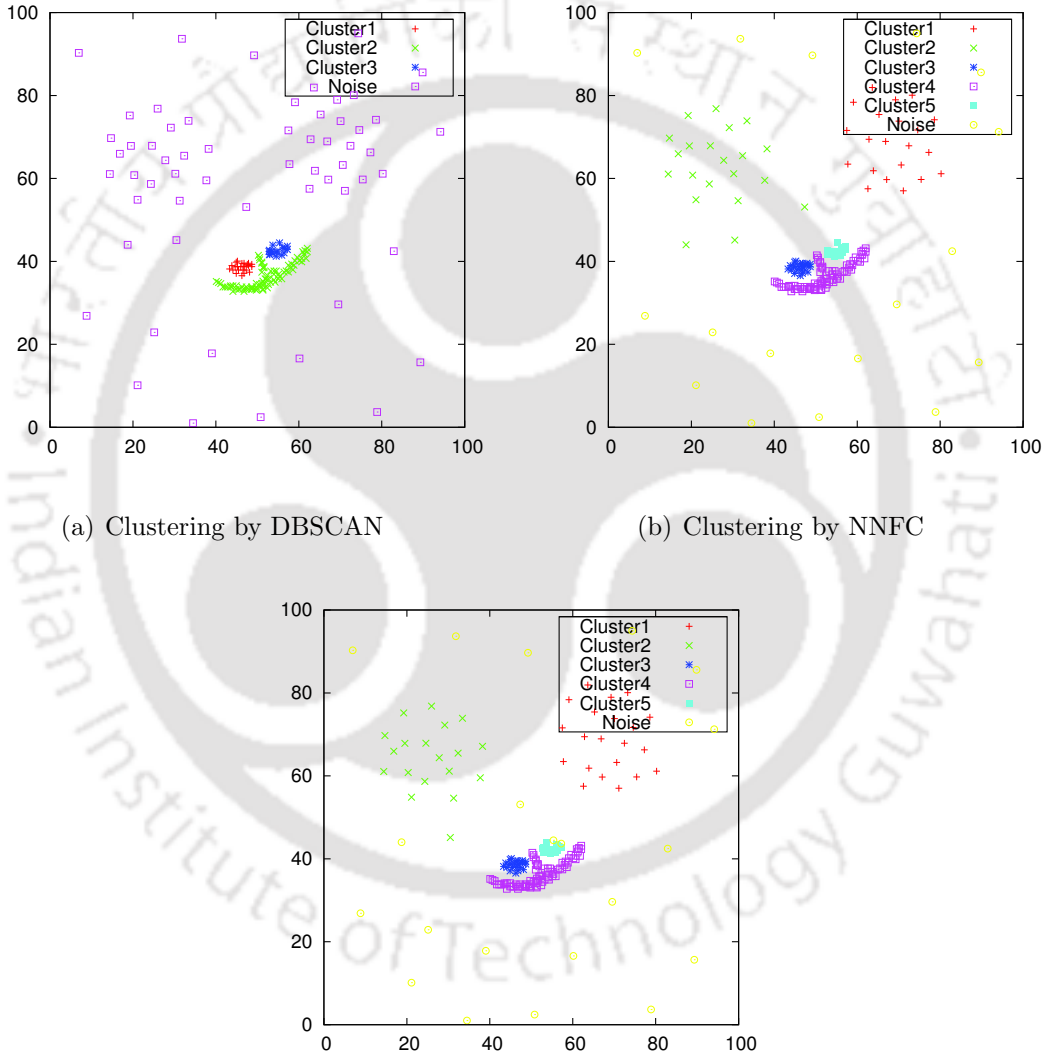
6.4 Performance Evaluations

To evaluate effectiveness, *NNFC* is implemented using C language on Intel(R) Core 2 Duo CPU (3.0 GHz) Desktop PC with 2 GB RAM. The method is tested with two synthetic and two real world dataset. Comparisons with DBSCAN and TI-K-Neighbor-Index supported NBC [Kryszkiewicz and Lasek (2010a)] are reported in this section. The *RI* and *Purity* are computed between output of a clustering method with the class labels of a given dataset.

Detailed description of experimental results and dataset used are given below. In our experiments, we consider the value of $\delta = 0.25$. However, slight change in value of δ does not effect on clustering quality.

6.4.1 Synthetic dataset

Variable Density Dataset [Zhou et al. (2005)]: A synthetic dataset is used which is similar to a dataset used in [Zhou et al. (2005)]. It is shown in Figure 6.2. This dataset consists of five clusters with variable densities. DBSCAN cannot find all five clusters with all possible values of parameters. With high density setting, DBSCAN can find only three clusters and treats other two as noise (Figure 6.3(a)). However, *NNFC* finds all five clusters as does TI-K-Neighbor-Index supported NBC (Figure 6.3(b), 6.3(c)).

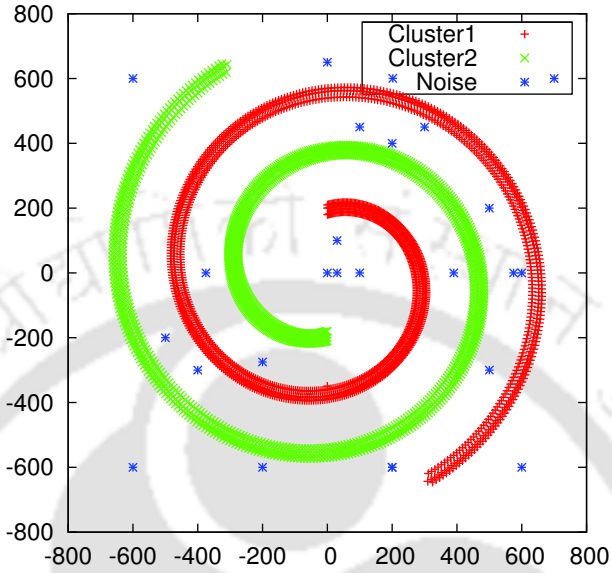


(a) Clustering by DBSCAN

(b) Clustering by NNFC

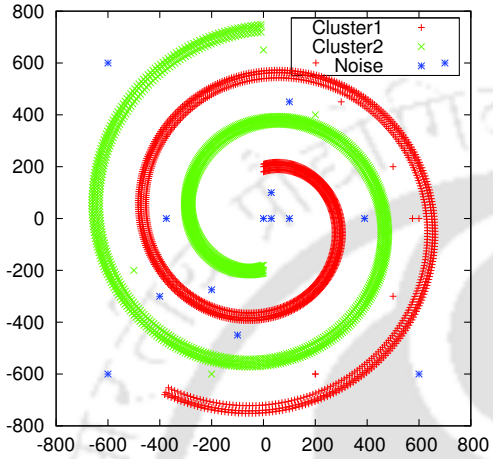
(c) Clustering by NBC

Figure 6.3: Clustering results with dataset [Zhou et al. (2005)]

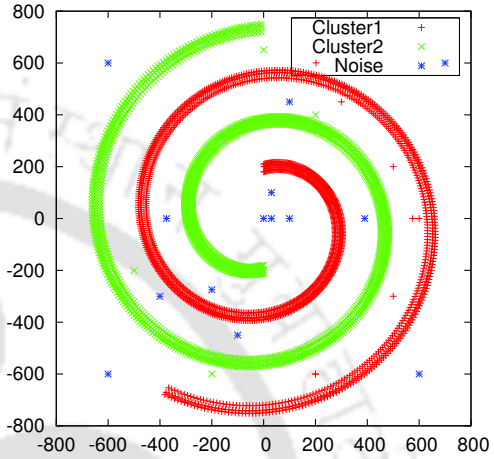


(a) Spiral with Outliers

Spiral Dataset with Outliers: Spiral dataset used in previous chapters is used here. However, 26 outlier points are added with it. This dataset has two spiral (arbitrary) shaped clusters with 3356 patterns as shown in Figure 6.4(a). From results of *NNFC* and TI-K-Neighbor-Indexed NBC with Spiral dataset, it is found that *NNFC* remains less sensitive to K values compared to TI-K-Neighbor-Indexed NBC method. Proposed method works well with a wide range of values ($K = 10$ to $K = 30$). It can detect two genuine clusters correctly with $K = 10$ to $K = 30$ (Figure 6.4). With $K = 10$, TI-K-Neighbor-Index supported NBC finds four clusters splitting each of the genuine clusters into two (Figure 6.5(a)). With $K = 15$, $K = 17$ and $K = 20$, it declares genuine cluster points as noise points (Figure 6.5). Clustering results and execution time of *NNFC* and TI-K-Indexed NBC are reported with $\delta = 0.25$ in Table 6.1.



(b) $K = 10, 15, 20$

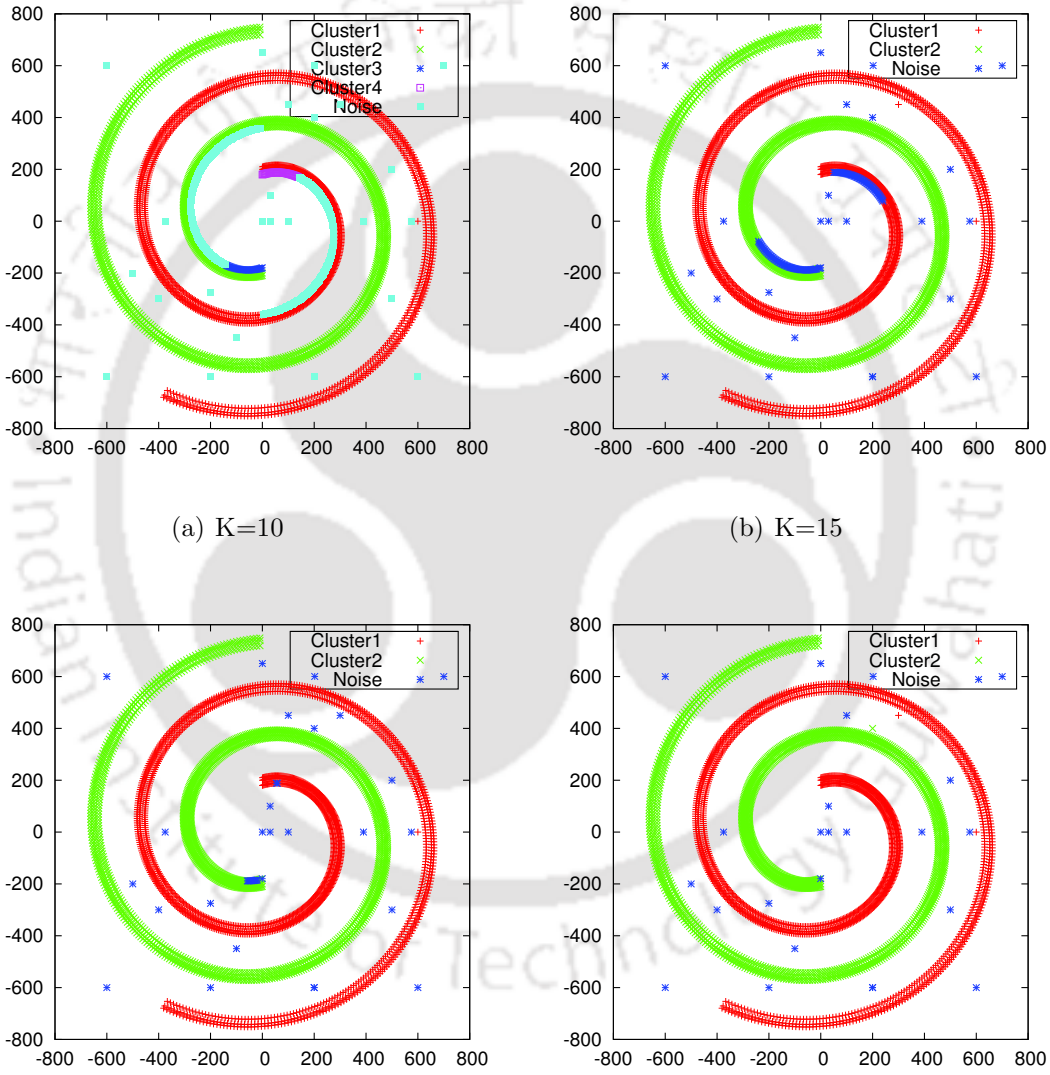


(c) $K = 25, 30$

Figure 6.4: Clustering produced by NNFC method with different values of K

Table 6.1: Experimental Results with Spiral Dataset

Value of K	Method	Time (in Seconds)	Purity	Rand Index	#Cluster
10	TI-K-Indexed NBC	0.91	0.951	0.899	4
	NNFC	**0.41**	**0.999**	**0.999**	**2**
15	TI-K-Indexed NBC	1.19	0.976	0.964	3
	NNFC	**0.43**	**0.999**	**0.999**	**2**
20	TI-K-Indexed NBC	1.48	0.998	0.998	3
	NNFC	**0.48**	**0.999**	**0.999**	**2**
25	TI-K-Indexed NBC	1.76	0.998	0.998	3
	NNFC	**0.55**	**0.998**	**0.998**	**2**



(a) K=10

(b) K=15

(c) K=17

(d) K=20

Figure 6.5: Clustering By NBC/TI-K-Indexed NBC with different K values

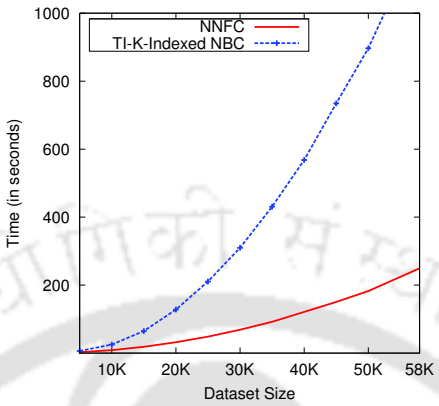


Figure 6.6: Execution Time of NNFC and TI-K-Indexed NBC Methods with Shuttle Data

6.4.2 Real world dataset

Shuttle Dataset: This dataset has 9 integer valued attributes of 58, 000 patterns distributed over 7 classes (after merging training and test sets). Class labels are eliminated from the all patterns. Clustering results produced by *NNFC* and TI-K-Index supported NBC method with different values of K are compared with class labels of the dataset using Rand Index and Purity measures and reported them in Table 6.2. It can be observed that clustering results ($RI = 0.854$) of *NNFC* method is better than the TI-K-Index supported NBC method ($RI = 0.585$) at value of $K = 30$. Execution time of TI-K-Neighborhood-Index supported NBC method and *NNFC* methods are also reported (Table 6.2). The number of clusters (7) produced by *NNFC* is close to the number of actual clusters (7) of the dataset with $K = 50$. Execution time of *NNFC* is more than five times faster than TI-K-Index supported NBC method for Shuttle dataset.

Figure 6.6 shows execution time of the *NNFC* and TI-K-Index supported NBC methods as data size varies from 5000 to 58000 for Shuttle dataset with $\delta = 0.25$. It may be noted that $NNFC$ consumes significantly less time as dataset size increases.

Page: This dataset has 5473 patterns distributed across five classes. A pattern corresponds to a page block of 54 documents. From experimental results, it can be noted that

Table 6.2: Results with Real World Dataset (UCI Machine Learning Repository)

Dataset Used	Value of K	Method	Time (in Seconds)	Purity	Rand Index	Number of Clusters
Shuttle	30	TI-K-Indexed NBC	1121.94	0.913	0.585	28
		NNFC	**241.46**	**0.921**	**0.854**	**11**
	35	TI-K-Indexed NBC	1225.92	0.914	0.674	18
		NNFC	**245.76**	**0.921**	**0.855**	**9**
	40	TI-K-Indexed NBC	1290.71	0.914	0.799	15
		NNFC	**250.25**	**0.915**	**0.812**	**8**
	50	TI-K-Indexed NBC	1467.50	0.913	0.801	13
		NNFC	**265.10**	**0.918**	**0.844**	**7**
Page	15	TI-K-Indexed NBC	2.73	0.924	0.747	13
		NNFC	**0.77**	**0.931**	**0.831**	**6**
	25	TI-K-Indexed NBC	3.41	0.925	0.779	10
		NNFC	**0.88**	**0.930**	**0.850**	**4**

NNFC produces better clustering (Purity = $0.931, 0.930$ and Rand Index= $0.831, 0.850$) than TI-K-Index supported NBC method (Purity = $0.924, 0.925$ and Rand Index= 0.747, 0.779). The *NNFC* takes less time compared to TI-K-Index supported NBC method (Table 6.2). With $K = 25$, clustering result ($RI = 0.850$) of *NNFC* is better than result ($RI = 0.779$) produced by the TI-K-Index supported NBC.

6.5 Conclusions

NNFC is a neighborhood based clustering method, which can find arbitrary shaped clusters in a dataset with variable densities. The *NNFC* calculates *Nearest Neighbor Factor* to locate relative position of a point in dataset. Based on position, a point is declared as a cluster point or noise point. Experimental results demonstrate that it outperforms TI-K-Index supported NBC in both clustering results and execution time.

Chapter 7

Conclusion and future scope of work

7.1 Summary of contributions of the thesis

Cluster analysis or clustering activity is to find groups of objects such that objects in a group are more *similar* than objects in different groups. Depending upon application domains, various clustering methods have been evolved over the years. These clustering methods can be categorized from different viewpoints. From *algorithmic viewpoint*, clustering methods are categorized into (*i*) *distance based methods* and (*ii*) *density based methods*. Distance based method uses only distance information between patterns in a given dataset to obtain clusters in the dataset, whereas density based method uses density information of each pattern in the dataset. This thesis proposed distance based as well as density based methods for finding "natural" clusters (clusters of arbitrary shapes and sizes) in large size dataset.

Distance based clustering methods usually find clusters of convex shapes in a dataset. However, distance based single-link can find clusters of arbitrary shapes and sizes. It scans a dataset multiples times and has time and space complexity of $O(n^2)$, where n is the size of the dataset. These are potentially severe drawbacks of clustering activity in large size data. To overcome these drawbacks, Chapter 3 of the thesis developed a hybrid clustering method, which combines two distance based methods- (*i*) leaders method, which needs a single dataset scan, and (*ii*) single-link with distance stopping

criteria. The hybrid method is suitable in large dataset and significantly faster than classical single-link method (distance stopping criteria) applied directly to the dataset. However, number of clusters produced by the hybrid method may be more compared to the number of clusters produced by that of the single-link method, *i.e.,* a cluster in a data may be divided into a number of sub-clusters by the hybrid method. This leads to error in clustering result. To handle this issue, as an enhancement to the proposed hybrid method, a correction step is added to the proposed hybrid method. Theoretical and experimental study demonstrated that clustering results provided by the correction step with hybrid method is exactly same as the results produced by that of the single-link method in significantly less time compared to the classical single-link method in large datasets.

Hybrid method and its enhancement method proposed in Chapter 3 produce a flat clustering of a dataset. These methods cannot be used to find nested arbitrary shaped clusters. To address this issue, a new data summarization scheme is proposed in Chapter 4. Proposed data summarization scheme generates a number of summarized units to capture summary of a dataset. The leaders clustering method is utilized to form summarized units. Classical hierarchical agglomerative (single-link) method is used to merge summarized units progressively. Experimental results illustrated that proposed summarization scheme outperforms data bubble and directional data bubble summarization schemes at execution time. Proposed summarization scheme combined with single-link produces better clustering hierarchy compared to data bubble based single-link method and the clustering hierarchy is very close to directional data bubble based single-link method.

Clustering results produced by data summarization scheme proposed in Chapter 4 deteriorates with high compression ratios. This approach cannot detect cluster separation ("gaps") present in a summarized unit. To address this issue, tolerance rough set theory based summarization scheme (*rough bubble*) is proposed in Chapter 5. Proposed *rough bubble* collects directional statistics of each summarized unit (also termed as *rough bubble*), which helps identifying cluster separation and assignment of patterns in proper rough bubbles. Experiments illustrated that tolerance rough set based summarization scheme

outperform directional data bubble in clustering results and execution time is marginally higher than directional data bubble.

Distance based approach cannot find arbitrary shaped clusters in the presence of outliers and varying density clusters in a dataset. To address this problem, a density based clustering method is proposed in Chapter 6. Proposed density based method uses K nearest neighbor information of a pattern to find relative 'importance' of the pattern in its neighborhood region. Depending upon 'importance' of a pattern, either it is added to a cluster (new or existing) or declared as outlier in a dataset. Experimental results demonstrated that proposed density based clustering method outperforms NBC and recently introduced TI-K-Neighborhood-Index supported NBC.

7.2 Scope for future work

The clustering methods and data summarization schemes developed in the thesis successfully addressed problems involved in clustering large size data. In this section, possible future extensions in these methods are discussed.

- Proposed *al*-SL method outputs a flat clustering of a given dataset. As an extension of this work, a method can be developed which would produce a dendogram. Higher levels of the dendogram would be similar to the dendogram produced by classical single-link method.

  Proposed *al*-SL method has a correction step, which makes this approach different from other hybrid clustering methods. The idea of correction step may be explored in other clustering methods and learning techniques in data mining.

  The proposed hybrid clustering methods (*l*-SL and *al*-SL) for arbitrary shaped clusters need a single input parameter h: distance between a pair of clusters in a given dataset. Final clustering outputs depend upon the value of h. At present, two different approaches are provided for finding suitable value of h in this thesis. The first approach fully depends upon domain knowledge and second approach on random sampling technique. One more approach can be added, which would use domain

knowledge as well as *weakest link* [Ben-David and Ackerman (2008)] to refine the value of h.

- Proposed summarization scheme *data sphere* was developed for speeding up the distance based single-link method. As one of extensions of this scheme, *data sphere* may be modified to work with density based hierarchical clustering method like OPTICS. The core distance, reachibility distance which are the basis of OPTICS method need to be redefined in *data sphere* scheme. Another research direction may be suitable modification of *data sphere* scheme to speed up DBSCAN, methods.

  Output of *data sphere* depends on scanning order of the dataset. A scheme can be developed to minimize the effect on outputs w.r.t. scanning order of the datasets.

- Proposed *rough bubble* summarization scheme addressed the problem of "gaps" present in a *rough bubble*. The scheme may be extended to address the following scenarios.

  - If distance between two rough bubbles is exactly 2τ and one bubble contains patterns from more than one clusters, then "gap" appears in the bubble.

  - If distance between two rough bubbles O_1 and O_2 is more than 2τ and there does not exist any bubble (say O_3) between O_1 and O_2 ($||O_3 - O_1/O_2|| <= ||O_1 - O_2||$), then cluster separation ("gaps") may belong to both bubbles. This underestimates distance between the bubble.

  The *rough bubble* scheme computes statistics of each rough bubble w.r.t. to all bubbles. Therefore, it makes a large number of distance computations. However, bubbles at considerable distance (far and away) do not influence clustering outputs. As an extension of the scheme, a technique can be developed, which would avoid collecting directional statistics of certain bubbles.

- An outlier detection method may be developed using parameter NNF proposed in Chapter 6. The detection method would find pre-specified numbers of top outliers

in a dataset. A mechanism may be introduced to avoid computations for certain genuine inlier points as focus is to find top outliers.

NNFC method uses TI-K-Neighborhood-Index for finding K nearest neighbors of a pattern quickly. As an extension of this work, usefulness of *Locality Sensitive Hashing* [Indyk and Motwani (1998)] may be studied in finding K nearest neighbors of each pattern and may be utilized in NNFC method.

- Proposed methods find arbitrary shaped clusters. As an extension of the work, *degree of arbitrariness* of clusters produced by these methods may be studied.
- Application of these methods in image segmentation, bio-informatics may be studied.

Bibliography

J. S. Aguilar-Ruiz and F. Azuaje. Knowledge Discovery in Lymphoma Cancer from Gene-Expression. In *Proceedings of the Intelligent Data Engineering and Automated Learning, IDEAL 2004*, pages 31–38, 2004.

I. S. Altingovde, R. Ozcan, H. C. Ocalan, F. Can, and Ö. Ulusoy. Large-scale cluster-based retrieval experiments on Turkish texts. In *Proceedings of the International ACM SIGIR conference on Research and development in information retrieval, SIGIR 2007*, pages 891–892, 2007.

I. S. Altingovde, E. Demir, F. Can, and Ö Ulusoy. Incremental cluster-based retrieval using compressed cluster-skipping inverted files. *ACM Transactions on Information Systems (TOIS)*, 26:1–36, June 2008.

M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. *Proceedings of the ACM International Conference on Management of Data, SIGMOD 1999*, 28(2):49–60, 1999.

D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the ACM-SIAM symposium on Discrete algorithms, SODA '07*, pages 1027–1035, 2007.

S. Asharaf and M. N. Murty. A rough fuzzy approach to web usage categorization. *Fuzzy Sets and Systems*, 148(1):119–129, 2004.

M. M. Astrahan. *Speech analysis by clustering, or the hyperphoneme method*. Technical Report, Stanford A.I. Project Memo, 1970.

F. Azuaje. Clustering-based Approaches to Discovering and Visualising Microarray Data Patterns. *Briefings in Bioinformatics*, 4(1):31–42, 2003.

F. Azuaje. A cluster validity framework for genome expression data. *Bioinformatics*, 18(2):319–320, 2002.

G. H. BALL and D. J. HALL. *Promenade–an on-line pattern recognition system*. Number RADC-TR-67-310. Technical Report, Stanford Research Institute, Stanford, Calif, 1967.

P. Bedi and S. Chawla. Use of Fuzzy Rough Set Attribute Reduction in High Scent Web Page Recommendations. In *Proceedings of the International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, RSFDGrC 2009*, pages 192–200, 2009.

S. Ben-David and M. Ackerman. Measures of Clustering Quality: A Working Set of Axioms for Clustering. In *Proceddings of Advances in Neural Information Processing Systems, NIPS 2008*, pages 121–128, 2008.

P. Berkhin. A Survey of Clustering Data Mining Techniques. *Grouping Multidimensional Data*, pages 25–71, 2006.

J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

N. Bolshakova, F. Azuaje, and P. Cunningham. An integrated tool for microarray data clustering and cluster validity assessment. In *Proceedings of the ACM Symposium on Applied Computing, SAC 2004*, pages 133–137, 2004.

F. H. Borgen and D. C. Barnet. Applying cluster analysis in counseling psychology research. *Journal of Counseling Psychology*, 34(4):456–468, 1987.

S. Boriah, V. Chandola, and V. Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008*, pages 243–254, 2008.

K. Bowyer and N. Ahuja. *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling Clustering Algorithms to Large Databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD 1998*, pages 9–15, 1998.

R. L. Breiger, S. A. Boorman, and P. Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology*, 12(3):328 – 383, 1975.

M. M. Breunig, H. P. Kriegel, and J. Sander. Fast Hierarchical Clustering Based on Compressed Data and OPTICS. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 2000*, pages 232–242, 2000.

M. M. Breunig, H.P. Kriegel, P. Krger, and J. S. Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2001*, pages 79–90, 2001.

K. Burbeck and S. Nadjm-Tehrani. ADWICE - Anomaly Detection with Real-Time Incremental Clustering. In *Proceedings of the International Conference on Information Security and Cryptology - ICISC 2004*, pages 407–424, 2004.

S.H. Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Survey*, 41(3), 2009.

V. Chaoji, M. Al Hasan, S. Salem, and M. J. Zaki. SPARCL: Efficient and Effective Shape-Based Clustering. In *Proceedings of the IEEE International Conference on Data Mining, ICDM 2008*, pages 93–102, 2008.

V. Chaoji, M. Al Hasan, S. Salem, and M. J. Zaki. SPARCL: an effective and efficient algorithm for mining arbitrary shape-based clusters. *Knowledge and Information Systems*, 21(2):201–229, 2009.

A. Chaturvedi, J. D. Carroll, P. E. Green, and J. A. Rotondo. A Feature-Based Approach to Market Segmentation Via Overlapping K-Centroids Clustering. *Journal of Marketing Research*, 34(3):370–377, 1997.

C. Y. Chen, Y. J. Oyang, and H. F. Juan. Incremental generation of summarized clustering hierarchy for protein family analysis. *Bioinformatics*, 20(16):2586–2596, 2004.

M. Dash, H. Liu, P. Scheuermann, and K. L. Tan. Fast hierarchical clustering and its validation. *Data & Knowledge Engineering*, 44(1):109–138, 2003.

S. Datta and S. Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4):459–466, 2003.

S. K. De and P. R. Krishna. Clustering web transactions using rough approximation. *Fuzzy Sets and Systems*, 148(1):131–138, 2004.

Y. Dinpashoh, A. Fakheri-Fard, M. Moghaddam, S. Jahanbakhsh, and M. Mirnia. Selection of variables for the purpose of regionalization of Iran's precipitation climate using multivariate methods. *Journal of Hydrology*, 297(1-4):109 – 123, 2004.

C. Dorai and A. K. Jain. Shape spectra based view grouping for free-form objects. In *Proceedings of International Conference on Image Processing, ICIP 1995*, pages III: 340–343, 1995.

W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing Flat Files Flatter. In *Proceedings of the International Conference on Knowledge Discovery and Data mining, KDD 1999*, pages 6–15, 1999.

J. C. Dun. A Fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

S. Dunn, L. Janos, and A. Rosenfeld. Bimean clustering. *Pattern Recognition Letters*, 1(1):169–183, 1983.

C. Elkan. Using the Triangle Inequality to Accelerate k-Means. In *Proceedings of the International Conference on Machine Learning, ICML 2003)*, pages 147–153, 2003.

M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining KDD-1996*, pages 226–231, 1996.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13 (1):3–16, 1981.

D. R. Furse, G. N. Punj, and D. W. Stewart. Individual search strategies in new automobile purchases. *Advances in Consumer Research*, 9:379–384, 1982.

K. C. Gowda and T. V. Ravi. Divisive clustering of Symbolic Objects using the concepts of both similarity and dissimilarity . *Pattern Recognition*, 28(8):1277–1282, February 1995.

S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. *Information Systems*, 26(1):35–58, 2001.

D. Gustafson and W. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. In *Conference on Decision and Control*, pages 761–776, 1979.

A. Guttman. R-trees: a dynamic index structure for spatial searching. In *13th ACM SIGMOD Int. Conf. Management Data*, volume 2, pages 47–57, Boston, MA, 1984.

J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

J. A. Hartigan. Consistency of Single Linkage for High-Density Clusters. *Journal of the American Statistical Association*, 76(374):388–394, 1981.

J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, 1975.

R. J. Hathaway and J. C. Bezdek. Recent convergence results for the fuzzy c-means clustering algorithms. *Journal of Classification*, 5:237–247, 1988.

M. A. Hearst and J. O. Pedersen. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96*, pages 76–84, 1996.

M. A. Hearst, D. R. Karger, and J. O. Pedersen. Scatter/Gather as a Tool for the Navigation of Retrieval Results. In *Proceedings of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, pages 65–71, 1995.

A. Hinneburg and H. H. Gabriel. DENCLUE 2.0: Fast Clustering Based on Kernel Density Estimation. In *Proceeding of the International Symposium on Intelligent Data Analysis, IDA 2007*, pages 70–80, 2007.

A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of International Conference on Knowledge Discovery and Data Mining, KDD 1998)*, pages 58–65, 1998.

T. B. Ho and N. B. Nguyen. Nonhierarchical document clustering based on a tolerance rough set model. *International Journal of Intelligent Systems*, 17:199–212, 2002.

H. Hruschka. Market definition and segmentation using fuzzy clustering methods. *International Journal of Research in Marketing*, 3(2):117–134, 1986.

N. Hubballi. *Design of Network Intrusion Detection Systems: An Effective Alarm Generation Perspective*. Ph. D. Thesis, Department of Computer Science and Engineering, IIT Guwahti, Guwahti, India, 2011.

P. Indyk and R. Motwani. Approximate nearest neighbors:towards removing the curse of dimensionality. In *Proceedings of 30th ACM Symposium on theory of Computing*, pages 604–613, 1998.

I. J. Jackson and H. Weinand. Classification of tropical rainfall stations: A comparison of clustering techniques. *International Journal of Climatology*, 15(9):985–994, 1995.

A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

A. K. Jain and H. C. Martin. Data Clustering: A User's Dilemma. In *Proceedings of Pattern Recognition and Machine Intelligence (PReMI)*, pages 1–10, 2005.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

A. K. Jain, Robert P. W. Duin, and J. Mao. Statistical Pattern Recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1): 4–37, 2000.

E. Januzaj, H. P. Kriegel, and M. Pfeifle. Towards Effective and Efficient Distributed Clustering. In *Proceedings of Workshop on Clustering Large Data Sets, ICDM 2003*, pages 49–58, 2003.

J. M. Jolion, P. Meer, and S. Bataouche. Robust Clustering with Applications in Computer Vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:791–802, 1991.

A. Joshi and R. Krishnapuram. Robust Fuzzy Clustering Methods to Support Web Mining. In *Proceeding of the Workshop on Data Mining and Knowledge Discovery, SIGMOD 1998*, pages 15–1, 1998.

A. Juan and E. Vidal. Comparison of Four Initialization Techniques for the k - Medians Clustering Algorithm. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition,*, pages 842–852, 2000.

G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *Journal on Scientific Computing*, 20:359–392, December 1998.

G. Karypis, E. H. Han, and V. Kumar. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *Computer*, 32(8):68–75, 1999.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data:an Introduction to Cluster Analysis*. John Wiley & Sons, USA, 1990.

S. Kawasaki, N. B. Nguyen, and T. B. Ho. Hierarchical Document Clustering Based on Tolerance Rough Set Model. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 2000*, pages 458–463, 2000.

B. King. Step-Wise Clustering Procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.

J. Kleinberg. An impossibility theorem for clustering. In *Proceeding of Advances in Neural Information Processing Systems, NIPS 2002*, pages 446–453, 2002.

H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing. *Knowledge and Information Systems*, 12(1):25–53, 2007.

A. Krause, J. Stoye, and M. Vingron. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, page 6:15, 2005.

M. Kryszkiewicz. Rough Set Approach to Incomplete Information Systems. *Information Sciences*, 112(1-4):39–49, 1998.

M. Kryszkiewicz and P. Lasek. A Neighborhood-based Clustering by Means of the Triangle Inequality. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2010*, pages 284–291, 2010a.

M. Kryszkiewicz and P. Lasek. TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality. In *Proceedings of the International Conference on Rough Sets and Current Trends in Computing, RSCTC 2010*, pages 60–69, 2010b.

P. Kumar, P. R. Krishna, R. S. Bapi, and S. K. De. Rough clustering of sequential data. *Data & Knowledge Engineering*, 63:183–199, 2007.

G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies 1. hierarchical systems. *The Computer Journal*, 9(4):373–380, 1967.

C. R. Lin and M. S. Chen. Combining Partitional and Hierarchical Algorithms for Robust and Efficient Data Clustering with Cohesion Self-Merging. *IEEE Trans. on Knowl. and Data Eng.*, 17(2):145–159, 2005.

T. Y. Lin and N. Cercone, editors. *Rough Sets and Data Mining: Analysis of Imprecise Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

P. Lingras and C. West. Interval Set Clustering of Web Users with Rough K-Means. *Journal of Intelligent Information Systems*, 23:5–16, 2004.

P. Lingras, M. Hogo, and M. Snorek. Interval set clustering of web users using modified Kohonen self-organizing maps based on the properties of rough sets. *Web Intelli. and Agent Sys.*, 2:217–225, August 2004.

M. Liu, X. Jiang, and A. C. Kot. A multi-prototype clustering algorithm. *Pattern Recogn.*, 42:689–698, May 2009.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129 – 137, 1982.

D. G. Lowe. Local Feature View Clustering for 3D Object Recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 682–688, 2001.

J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

P. C. Mahalanobis. On the Generalized Distance in Statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

G. Martelet, C. Truffert, B. Tourlire, P. Ledru, and J. Perrin. Classifying airborne radiometry data with Agglomerative Hierarchical Clustering: A tool for geological mapping in context of rainforest (French Guiana). *International Journal of Applied Earth Observation and Geoinformation*, 8(3):208 – 223, 2006.

G. M. Mimmack, S. J. Mason, and J. S. Galpin. Choice of distance matrices in cluster analysis: Defining regions. *Journal of Climate*, 14(12):2790–2797, 2001.

S. Mitra and T. Acharya. *Data Mining: Multimedia, Soft Computing, and Bioinformatics*. John Wiley, USA, 2003.

F. Murtagh. Complexities of hierarchic clustering algorithms: state of the art. *Computational Statistics Quarterly*, 1:101–113, 1984.

M. N. Murty and G. Krishna. A hybrid clustering procedure for concentric and chain-like clusters. *International Journal of Computer and Information Science*, 10 (6):397–412, 1981.

M. N. Murty and G. Krishna. A computationally efficient technique for data-clustering. *Pattern Recognition*, 12(3):153 – 158, 1980.

M. Nanni. Speeding-Up Hierarchical Agglomerative Clustering in Presence of Expensive Metrics. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2005*, pages 378–387, 2005.

S. Nassar, J. Sander, and C. Cheng. Incremental and Effective Data Summarization for Dynamic Hierarchical Clustering. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2004*, pages 467–478, 2004.

M. Nei and S. Kumar. *Molecular Evolution and Phylogenetics*. Oxford University Press, Oxford,USA, 2000.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems, NIPS 2001*, pages 849–856, 2001.

R. T. Ng and J. Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, September 2002.

C. F. Olson. Parallel Algorithms for Hierarchical Clustering. *Parallel Computing*, 21:1313–1325, 1995.

Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

L. Polkowski, A. Skowron, and J. Zytkow. Tolerance Based Rough Sets. In *Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management*, pages 55–58, 1995.

R. Porter and C. N. Canagarajah. A robust automatic clustering scheme for image segmentation using wavelets. *IEEE Transactions on Image Processing*, 5(4):662–665, 1996.

G. Punj and D. W Stewart. Cluster Analysis in Marketing Research: Review and Suggestions for Application. *Journal of Marketing Research*, 20(2):134–148, 1983.

W. M. Rand. Objective Criteria for Evaluation of Clustering Methods. *Journal of American Statistical Association,*, 66(336):846–850, 1971.

A. Rosenfeld, V. B. Schneider, and M. K Huang. An application of cluster detection to text and picture processing. *IEEE Transactions on Information Theory*, 15(6): 672–681, 1969.

E. H. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22 – 32, 1969.

E. Schikuta. Grid-clustering: a fast hierarchical clustering method for very large data sets. In *Proceedings the International Conference on Pattern Recognition, ICPR 1996*, pages 101–105, 1996.

M. J. Shaw, C. Subramaniam, G. W. Tan, and M. E. Welge. Knowledge management and data mining for marketing. *Decision Support Systems*, 31(1):127 – 137, 2001.

A. Skowron and J. Stepaniuk. Tolerance Approximation Spaces. *Fundamenta Informaticae*, 27:245–253, 1996.

D. Ślezak and P. Wasilewski. Granular Sets — Foundations and Case Study of Tolerance Spaces. In *Proceedings of the International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, RSFDGrC 2007*, pages 435–442, 2007.

R. Slowinski and D. Vanderpooten. A Generalized Definition of Rough Approximations Based on Similarity. *IEEE Transactions on Knowledge and Data Engineering*, 12:331–336, 2000.

P. H. Sneath and A. Sokal. *Numerical Taxonomy*. Freeman, London, 1973.

H. Spath. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*. Ellis Horwood, UK, 1980.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communication of ACM*, 29(12):1213–1228, December 1986.

J. L. Starck and F. Murtagh. *Astronomical Image and Data Analysis*. Springer-Verlag, Berlin Heidelberg, Germany, 2002.

H. Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci. Cl. III.*, 4:801–804 (1957), 1956.

P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Pearson Education,Inc. and Dorling Kindersley Publishing, Inc., New Delhi, India, 2009.

A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. C. Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22: 2405–2412, September 2006.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition, third ed.* Academic Press, Inc., Orlando, 2006.

P. Trahanias and E. Skordalakis. An efficient sequential clustering method. *Pattern Recognition*, 22(4):449–453, 1989.

J. P. Tremblay and R. Manohar. *Discreate Mathematical Structures with Applications to Computer Science*. Tata McGraw-Hill Publishing Company Limited, New Delhi, 1997.

Y. Unal, T. Kindap, and M. Karaca. REDEFINING THE CLIMATE ZONES OF TURKEY USING CLUSTER ANALYSIS. *International Journal of Climatology*, 23:1045–1055, 2003.

P. A. Vijaya, M. N. Murty, and D. K. Subramanian. Efficient bottom-up hybrid hierarchical clustering techniques for protein sequence classification. *Pattern Recognition*, 39(12):2344–2355, 2006.

N. X. Vinh and J. Epps. A Novel Approach for Automatic Number of Clusters Detection in Microarray Data Based on Consensus Clustering. In *Proceedings of IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2009*, pages 84–91, 2009.

A. G. Wacker. *A cluster approach to finding spatial boundaries in multispectral imagery*. Number 122969. Laboratory for Applications of Remote Sensing Information Note, Purdue University, 1969.

R. Weber, H. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of International Conference on Very Large Data Bases, VLDB'98*, pages 194–205, 1998.

M. A. Wong. A hybrid clustering algorithm for identifying high density clusters. *Journal of the American Statistical Association*, 77(380):841–847, 1982.

R. Xu and D. Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965.

H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceedings of the ACM SIGIR conference on Research and development in information retrieval, SIGIR 2004*, pages 210–217, 2004.

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the International Conference on Management of Data, SIGMOD 1996*, pages 103–114, 1996.

Y. Zhao and G. Karypis. Criterion Functions for Document Clustering: Experiments and Analysis. Technical report, University of Minnesota, 2002.

J. Zhou and J. Sander. Data bubbles for non-vector data: speeding-up hierarchical clustering in arbitrary metric spaces. In *Proceedings of the Conference on Very Large Databases, VLDB '2003*, pages 452–463, 2003.

S. Zhou, Y. Zhao, J. Guan, and J. Z. Huang. A Neighborhood-Based Clustering Algorithm. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2005*, pages 361–371, 2005.

Publications out of the work

International Journals

1. **Bidyut Kr. Patra**, Sukumar Nandi and P. Viswanath *A distance based clustering method for arbitrary shaped clusters in large datasets*, **Pattern Recognition (Elsevier)**, Vol. 44, Issue 12, December, 2011, Page 2862-2870 **(2011)**

2. **Bidyut Kr. Patra** and Sukumar Nandi *Tolerance Rough Set Theory based Data Summarization for Clustering Large Datasets*, **Transaction on Rough Sets (Springer)**, Vol XIV, August, 2011, Page 139-158 **(2011)**

International Conference

1. **Bidyut Kr. Patra** and Sukumar Nandi *Neighborhood based Clustering Method for Arbitrary Shaped Clusters* In Proceeding of the 19^{th} International Symposium on Methodologies for Intelligence System, **(ISMIS 2011)**, June 28-30, 2011 **Warsaw University of Technology, Warsaw, Poland**. (LNCS) **(2011)**

2. **Bidyut Kr. Patra**, Neminath Hubballi, Santosh Biswas, Sukumar Nandi *Distance based fast hierarchical clustering method for large datasets* In Proceeding of the Seventh International Conference on Rough Sets and Current Trends in Computing **(RSCTC 2010)**, June 28-30, 2010, **University of Warsaw, Warsaw, Poland**. (LNAI) **(2010)**

3. **Bidyut Kr. Patra**, Sukumar Nandi *A Fast Single-link Clustering Method based on*

a Tolerance Rough Set Model, In Proceeding of the Twelfth International Conference on Rough Sets, Fuzzy Sets, Data Mining & Granular Computing **(RSFDGrC 2009)**, December 16-18, 2009, **IIT Delhi, India**. (LNAI) **(2009)**

4. **Bidyut Kr. Patra**, Sukumar Nandi, and P.Viswanath *Data Summarization based First Hierarchical Clustering Methods for Large Datasets*, IEEE International Conference on Information Management and Engineering **(ICIME 2009)**, **Kuala Lumpur**, April, 2009 (IEEE) **(2009)**

5. **Bidyut Kr. Patra**, P.Viswanath and Sukumar Nandi, *A Fast Single Link Clustering Method for Large Data Sets*, MCDES **(Centenary Conference)**, May 2008, **Indian Institute of Science , Bangalore, India (2008)**

Other publications of the author

Book Chapter

1. P.Viswanath, **Bidyut Kr. Patra** and V.Suresh Babu, *Some Efficient and Fast Approaches to Document Clustering*, appears as a book chapter in "Handbook of Research on Text and Web Mining Technologies", Information science reference (IGI Global), Hershey, New York **(2008)**

International Conference

- Neminath Hubballi, **Bidyut Kr. Patra**, Sukumar Nandi *NDoT: NDoT: Nearest Neighbor Distance Based Outlier Detection Technique* to appear in The 4^{th} International Conference on Pattern Recognition and Machine Intelligence (**PReMI 2011**), June 27 - July 1, 2011, **Higher School of Economics, Moscow, Russia** . **(2011)**

Curriculum Vitae

Name : Bidyut Kumar Patra

Father's Name : Krishna Chandra Patra

Address : Department of Computer Science & Engineering,
IIT Guwahati, Guwahti, Assam, INDIA

Email : bidyut@iitg.ernet.in

Bidyut Kumar Patra obtained his B.Sc. (Physics (Hons.)), B.Tech. (CSE), and M.Tech. (CSE) from University of Calcutta, Kolkata, India in 1996, 1999 and 2001, respectively. He has been working for his Ph.D. at the Indian Institute of Technology Guwahati (IIT Guwahati), Guwahati, India since July, 2005. He was a regular research scholar at IIT Guwahati during July, 2005-Febuary, 2010. He was awarded CSIR-Senior Research Fellowship by CSIR, Govt. of India, New Delhi, India during January, 2009-Febuary, 2010. Prior to joining IIT Guwahati, he was a faculty in the Department of Computer Science & Engineering of Haldia Institute of Technology, Haldia, India for three and half years. He had also a short stint at Tezpur Central University as an Assitant Professor during March, 2010-September, 2011. Presently, he is an Assistant Professor in the Deparment of Computer Science & Engineering of National Institute of Technology Rourkela , Rourkela, Orissa, India. His research interests include Pattern Recognition, Data Mining and Algorithms. He has authored 11 research articles till date.

কত প্রদীপ এই থালাতে সাজিয়েছিলে আপন হাতে--

কত যে তার নিবল হাওয়ায়, পৌঁছল না চরণছায়ে॥