INVESTIGATIONS IN FUZZY BASED LEARNING ALGORITHMS WITH APPLICATION TO BIG DATA CLASSIFICATION

A THESIS

submitted in partial fulfillment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

NEHA BHARILL

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JULY 2017

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **INVESTIGATIONS IN FUZZY BASED LEARNING ALGORITHMS WITH APPLICATION TO BIG DATA CLASSIFICATION** in the partial fulfillment of the requirement for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING**, **Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from JANUARY, 2013 to JULY, 2017 under the supervision of Dr. Aruna Tiwari, Associate Professor, Discipline of Computer Science & Engineering, Indian Institute of Technology Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the Student with date
(NEHA BHARILL)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of the Thesis Supervisor with date
(Dr. ARUNA TIWARI)

NEHA BHARILL has successfully given her Ph.D. Oral Examination held on **15-01-2018**.

Signature of Chairperson (OEB)	Signature of External Examiner	Signature of Thesis Supervisor
Date:	Date:	Date:
Signature of PSPC Member #1	Signature of PSPC Member #2	Signature of Convener, DPGC
Date:	Date:	Date:

Signature of Head of Discipline

Date:

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. At first, I would like to thank my supervisor Dr. Aruna Tiwari, who was a constant source of inspiration during my work, without her constant guidance and research directions this research work could not be completed. Her continuous support and encouragement has motivated me to remain streamline in my research work. I am grateful to Dr. Kapil Ahuja, for all his extended help, support, and constructive feedback. I am also grateful to Dr. Surya Prakash, HOD of Computer Science for his constructive feedback.

I am thankful to Dr. Ram Bilas Pachori and Dr. Surya Prakash, my research committee member for taking out some valuable time to evaluate my progress all these years. Their good comments and suggestions helped me to improve my work at various stages.

I wish to thank all my colleagues, and staff from the Discipline of Computer Science and Engineering for thier suggestions and friendship. I extend my sincere thanks to many government bodies like DST, CSIR, MPCST apart from IIT Indore to help me with financial support to attend international conferences, which gave me a right platform at the right time and helped a lot to groom my research work.

I would like to thank my family, especially my mother, father, and my husband for always believing in me, for their continuous source of inspiration and their support in my decisions. Without whom I could not have made it here. I also want to thank my in-laws for their support and blessings.

Dated: (Neha Bharill)

Dedicated to my parents

Abstract

Clustering is one of the most widely used methods for exploratory data analysis. The need of clustering arises in many real-life problems, such as gene analysis, image processing, text organization, community detection, disease diagnosis, and protein categorization. Fuzzy clustering is one of the most widely used methods to handle such real-life problems. The principle advantage of fuzzy clustering is that the membership degrees expresses how ambiguously a data point should belong to a cluster. However, there are many aspects of the design of fuzzy clustering need to be addressed for improving the overall performance of fuzzy clustering by preserving the quality of clustering to handle different categories of data, such as Very Large (VL) and Big Data. The quality of clustering is affected by many parameters during the clustering process. One such parameter is the number of clusters (c), that can be determined during clustering. Apart from this, two other parameters are the fuzzifier and the locations of cluster centers that need to be selected appropriately for better efficacy of fuzzy clustering. Furthermore, looking towards the current need of clustering processes to handle VL and Big Data in real-life situations, there is a need to model incremental and scalable clustering algorithms.

The performance improvement of fuzzy clustering has been done by proposing a novel cluster validity index, which incorporates intra-cluster compactness, inter-cluster separation, and inter-cluster overlap measures to find the optimal number of clusters in a dataset. Further, for the appropriate selection of other parameters of fuzzy clustering, i.e., the fuzzifier and the location of cluster centers, we proposed two variants of hybrid fuzzy clustering algorithms. Firstly, the proposed quantum-inspired evolutionary fuzzy algorithm for data clustering, which uses quantum computing principle to explore a large search space for an appropriate selection of a fuzzifier parameter in fuzzy clustering. Sec-

ondly, the enhanced quantum-inspired evolutionary fuzzy clustering algorithm is proposed, which explores a large search space for the appropriate selection of all parameters of fuzzy clustering, i.e., the number of clusters, the location of cluster centers, and the fuzzifier.

In order to handle different categories of data, such as VL and Big Data, other novel approaches for fuzzy clustering are designed. Firstly, we proposed an incremental clustering algorithm integrated with a supervised classification method for processing VL datasets. The proposed method processes the VL dataset in chunks and sequentially performs clustering of each chunk. It removes the problem of loading the entire data in memory all at once by reducing the run-time and shows significant improvement in classification accuracy. To handle Big Data, scalable clustering algorithms are proposed and implemented using the Apache Spark framework which provides an in-memory computation capability for proceeding with faster computations on Big Data. The proposed approach reduces run-time and space complexity. It also shows competitive results in terms of various performance measures.

The proposed scalable clustering algorithm is applied to a real-life Big Data Problem. For this, a massive protein database of a complex plant genome, which is of size 80 GB, has been collected from the Directorate of Soybean Research (DSR), Indore under the Indian Council of Agricultural Research (ICAR). The collected protein database is used for the categorizing of protein sequences into the respective superfamilies. This categorization helps to assist DSR scientists in enhancing the productivity of next generation protein sequences of different species of plant. To do this, first, a rigorous study and analysis of protein sequences of the complex plant genome are carried out. Then, a novel feature extraction approach is designed that extracts fixed-length numeric feature vectors consisting of only six dimensions to represent a long chain of protein sequences. This feature extraction method is applied to the real-life protein database for its preprocessing and then the preprocessed data is passed as input to the scalable clustering algorithm for efficient categorization of protein sequences into superfamilies. The exhaustive results on this massive protein database are evaluated in terms of various performance measures.

List of Publications

(A) International Journal

(i) Published

[1] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy based scalable clustering algorithms for handling big data using apache spark," IEEE Transactions on Big Data, vol. 2, no. 4, pp. 339-352, 2016.

[2] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O.P. Patel, A. Tiwari, M. Joo, W. Ding, and C.T. Lin, "A review of clustering techniques and developments," Neurocomputing, Elsevier, vol. 267, pp. 664-681, 2017.

[3] N. Bharill, O.P. Patel, and A. Tiwari, "Quantum-inspired Evolutionary Approach for Selection of Optimal Parameters of Fuzzy Clustering," International Journal of System Assurance Engineering and Management, Springer, vol. 8, pp. 1-13, 2017.

(ii) Communicated (Under review)

[1] N. Bharill, O.P. Patel, A. Tiwari, M. Prasad, and C.T. Lin, "A generalized enhanced quantum fuzzy approach for efficient data clustering," IEEE Transactions on Emerging Topics in Computational Intelligence (Revision Submitted - Nov. 20, 2017, Current Status - Under Review).

[2] N. Bharill, A.Tiwari, A. Malviya, A. Gupta, D. Puthal, and A. Saxena, "Fuzzy knowledge based performance analysis on big data," Neurocomputing, Elsevier (Submitted - Nov. 2, 2017, Current Status - Under Review).

[3] N. Bharill, and A. Tiwari, "Protein superfamily classification using a novel feature extraction approach," International Journal of Data Mining and

Bioinformatics (IJDMB), Inderscience (Submitted - Aug. 8, 2017, Current Status Under Review).

(B) Book Chapters

[1] N. Bharill and A. Tiwari, "Handling big data with fuzzy based classification approach," in Proc. of Third Annual World Conference on Soft Computing, Advance Trends in Soft Computing, ser. Studies in Fuzziness and Soft Computing, Springer International Publishing, 2014, vol. 312, pp. 219-227.

[2] N. Bharill and A. Tiwari, "A novel technique of feature extraction based on local and global similarity measure for protein classification," in Proc. of International Conference on Bioinformatics Models, Methods and Algorithms, Bioinformatics, Scitepress Digital Library, 2015, vol. 1, pp. 219-224.

(C) International Conferences

[1] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy based clustering algorithms to handle big data with implementation on apache spark," in Proc. of 2016 Second IEEE International Conference on Big Data Computing Service and Applications, IEEE BigDataService, IEEE, Exeter College, Oxford, UK, March, 2016, pp. 95-104. (**IEEE Computer Society Conference**, Link: http://www.big-dataservice.net/).

[2] N. Bharill, O.P. Patel, and A. Tiwari, "An enhanced quantum-inspired evolutionary fuzzy clustering," in Proc. of 2015 IEEE Symposium Series on Computational Intelligence, IEEE, Cape Town, South Africa, December, 2015, pp. 772 - 779 (**Flagship Conference**, Link: http://ieee-ssci.org.za:8080/)

[3] O.P. Patel, N. Bharill, and A. Tiwari, "A quantum-inspired fuzzy based evolutionary algorithm for data clustering," in Proc. of 2015 IEEE International Conference on Fuzzy Systems, IEEE, Istanbul, Turkey, August,

2015, pp. 1 - 8 (**RANK A**, Link: http://202.141.161.27/ qiang/cs-conf-ranking.htm)

[4] N. Bharill, A. Tiwari, and A. Rawat, "A novel technique of feature extraction with dual similarity measures for protein sequence classification," in Proc. of 2014 International Conference on Computer, Communication and Convergence, Procedia Computer Science, Elsevier, Bhubaneswar, India, December, 2014, vol. 48, pp. 795-801.

[5] N. Bharill and A. Tiwari, "Enhanced cluster validity index for the evaluation of optimal number of clusters for fuzzy c-means algorithm," in Proc. of 2014 IEEE International Conference on Fuzzy Systems, IEEE World Congress on Computational Intelligence, IEEE, Beijing, China, July, 2014, pp. 1526 - 1533 (**RANK A**, Link: http://202.141.161.27/ qiang/cs-conf-ranking.htm).

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

FCM	Fuzzy C-Means
VL	Very Large
LFCM	Literal Fuzzy C-Means
spFCM	single-pass Fuzzy C-Means
oFCM	online Fuzzy C-Means
brFCM	bit-reduced Fuzzy C-Means
RSIO-FCM	Random Sampling Iterative Optimization Fuzzy C-Means
GB	GigaByte
DSR	Directorate of Soybean Research
ICAR	Indian Council of Agricultural Research
TBs	TeraBytes
QIE-FCM	Quantum-inspired Evolutionary Fuzzy Clustering
EQIE-FCM	Enhanced Quantum-inspired Evolutionary Fuzzy Clustering
NMI	Normalized Mutual Information
ARI	Adjusted Rand Index
SRSIO-FCM	Scalable Random Sampling with Iterative Optimization Fuzzy C-Means
CPU	Central Processing Unit
RAM	Random Access Memory
GHz	GigaHertz
rseFCM	random sampling plus extension Fuzzy C-Means
SLFCM	Scalable Literal Fuzzy C-Means
SrseFCM	Scalable random sampling plus extension Fuzzy C-Means
CPSF	Co-occurrence based Probability Specific Feature Extraction
PAM	Partition Around Medoids
CLARANS	Clustering Large Applications based on RANdomized Search

CLARA	Clustering LARge Applications
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
CURE	Clustering Using REpresentatives
ROCK	RObust Clustering using linKs
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points To Identify the Clustering Structure
DBCLASD	Distribution Based Clustering of Large Spatial Databases
DENCLUE	DENsity-based CLUstEring
STING	STatistical INformation Grid
CLIQUE	Clustering In QUEst
OptiGrid	Optimal Grid Partitioning
EM	Expectation Maximization
SOMs	Self Organizing Maps
OSI	Overlap and Separation Index
PBMF	Pakhira-Bandhyopadhya-Maulik
PCAES	Partition Coefficient and Exponential Separation
QM-FCM	Quantum-Modeled Fuzzy C-Means
RQECA	Real-parameter Quantum-inspired Evolutionary Clustering Algorithm
RQIEA	Real-parameter Quantum-inspired Evolutionary Algorithm
QEFCM	Quadratic Entropy based Fuzzy C-Means
VGA	Variable String Length Genetic Algorithm
FPSO	Fuzzy Particle Swarm Optimization
PSO	Particle Swarm Optimization
mrFCM	multistage random Fuzzy C-Means
psFCM	partition simplification Fuzzy C-Means
geFFCM	generalized extensible Fast Fuzzy C-Means
HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Datasets
YARN	Yet Another Resource Negotiator
I/O	Input output
API	Application Program Interface
SQL	Structured Query Language
SIMR	Spark in MapReduce

SPMD	Single Program Multiple Data
MPI	Message Passing Interface
SVD	Singular Value Decomposition
UCI	University of California, Irvine
FCMVGA	Fuzzy C-Means clustering based on Variable String Length Genetic Algorithm
PID	Pima Indian Diabetes
AMOGA	Adaptive Multi-objective Genetic Algorithm
RBFN	Radial Basis Function Network
PCA	Principal Component Analysis
SVM	Support Vector Machine
NB	Naive Bayes
k-NN	k-Nearest Neighbors
BLTA	Boolean-Like Training Algorithm
WEKA	Waikato Environment for Knowledge Analysis
SD	Standard Deviation
DNA	Deoxyribonucleic Acid
SNP	Single-nucleotide polymorphism

Chapter 1

Introduction

Clustering is an approach for exploratory data analysis that collects data points into groups called clusters, such that data points lying within the same group are more similar to each other than to the data points lying in different groups [1–3]. The formed clusters are appropriate for the exploration of the underlying structure of the data and the better understanding of the nature of data [4–6]. Clustering is also known as unsupervised learning and it depends on the kind of clustering algorithm being utilized [1, 7–9]. Clustering is used in a wide variety of applications such as gene analysis, image processing, text organization, community detection, disease diagnosis, and protein categorization [10–15]. In real-life situations, it's hard to capture the underlying structure of data with a single clustering algorithm because every dataset has a different geometrical distribution in feature space. Also capturing the underlying structure of data with a single clustering algorithm becomes much harder when there exists an ambiguity in designating a data point to a particular group or cluster. Therefore, there is an immense need to develop different clustering approaches that can adequately capture the underlying structure of various datasets into groups.

Clustering algorithms are broadly divided into two groups: hierarchical clustering and partitioning clustering [16, 17]. Hierarchical clustering algorithms yield a complete hierarchy by decomposing a dataset into several levels of nested partitions (clusters) represented by a dendrogram (tree), whereas the partitioning clustering algorithm attempts to partition the dataset into some prespecified number of clusters, usually by optimizing an objective function. Partitioning based clustering algorithms are broadly classified into hard (or crisp) clustering

and fuzzy clustering [18]. Hard clustering algorithms partition the data points into a set of disjoint clusters or groups, where each data point belongs to only one group. The major disadvantage of hard clustering is that it becomes inappropriate for real datasets in which there are no definite boundaries between the clusters. Thus, people realized the need to handle this situation that gave birth to fuzzy based algorithms. After the introduction of fuzzy theory by Lotfi Zadeh [19], the problem of handling the indefinite boundaries among the clusters became easier. Thus, this gave birth to the fuzzy clustering algorithms. A fuzzy clustering algorithm furnishes fuzzy partitions to measure the uncertainty that a data point belongs to two or more clusters with a partial degree of membership [20]. The principle advantage of fuzzy clustering is that the membership degrees expresses how ambiguously a data point should belong to a cluster. Furthermore, these membership degrees offer a means for handling ambiguous data appropriately [21].

Inspired by fuzzy set theory, Bezdek [22] proposed one of the most popular and widely used fuzzy clustering algorithms known as FCM. It partitions the collection of n data points into c fuzzy groups such that an objective function of a dissimilarity measure is minimized. With this technique, a data point can be grouped into more than one cluster. Thus, fuzzy clustering is very flexible in handling uncertainty, so it helps in solving many real-life complex problems [23].

Despite the wide acceptance of FCM, it has several flaws. In standard FCM, the selection of the weighted exponent or fuzzifier (m), the number of clusters (c), and the set of initial cluster centers (V) play a significant role in fuzzy cluster analysis. Moreover, FCM is very sensitive to the selection of these parameters as they have a direct impact on the formation of final clusters [20, 24–26]. These parameters are randomly selected in FCM, due to which FCM does not guarantee unique clustering results and makes the iterative algorithm to fall into a local optimal solution [26, 27]. Another major issue with the FCM is that its run-time exponentially increases with an increase in the number of data points in the dataset. Thus, the larger the size of a dataset, the longer the clustering algorithm will take to produce the partitions. In addition to this, processing the entire dataset with standard FCM becomes infeasible even in the case of large and VL datasets.

According to Huber statistics [28], dataset are classified into various cate-

Table 1.1: Huber's description of dataset sizes

Bytes	10^6	10^8	10^{10}	10^{12}	$10^{>12}$
Sizes	medium	large	huge	monster	VL

gories based on different sizes, as given in Table 1.1. Bezdek and Hathway [29] added the category of VL data to this table. There are various types of fuzzy based clustering algorithms that have been proposed to cluster VL data [29–36]. Literal schemes simply perform clustering on the entire dataset [37]. On the contrary, extended fuzzy clustering approaches apply a clustering algorithm to a sample of the full dataset using LFCM [36,37], and then non-iteratively extend the sample clustering results to obtain clusters for the remaining data. Some algorithms include their noniterative mechanisms for extension of clustering results on the remaining data, referred to as extensible algorithms [29, 32, 36]. Other leading algorithms include spFCM [33] and oFCM [38], which are incremental algorithms to compute an approximate FCM solution. The brFCM [39] algorithm uses a binning strategy for data reduction. These approaches adopt different strategies to perform clustering of VL data. Despite the rapid development of clustering algorithms aimed at handling VL data, there is a lack of adoption of these techniques in the wider data mining and other application communities dealing with Big Data problems [40, 41]. A likely reason for this is that these algorithms are not scalable to handle Big Data [40, 41]. This prevents many state-of-the-art clustering algorithms from being widely applied at Big-Learning scales [40, 41].

There are many aspects of the design of fuzzy clustering that need to be addressed for improving its performance to handle Big Data while preserving the quality of clustering results. In fuzzy clustering algorithms, the number of clusters is randomly decided by trial and error by the user for clustering of a dataset. The problem with the random selection of the number of clusters is that the inappropriate selection of the number of clusters is unable to capture the underlying structure of data points accurately, and suffers from a local convergence problem. Therefore, we propose a novel cluster validity index for fuzzy clustering to identify the optimal number of clusters for different datasets. It is observed that other soft computing methods [27, 42–44] are also integrated with fuzzy clustering to improve its overall performance. Therefore, we have investi-

gated the use of quantum computing in fuzzy clustering and propose two novel hybrid clustering algorithms for finding optimal parameters of fuzzy clustering from a large search space. Furthermore, to accelerate the speed of fuzzy clustering to handle VL datasets, we propose an incremental clustering algorithm called RSIO-FCM integrated with supervised classification mechanism. In addition to this, we also propose scalable fuzzy based iterative clustering algorithms to handle Big Data. The proposed incremental and scalable clustering algorithms are tested on various datasets. In addition to this, to investigate the performance of the scalable clustering algorithm on a real-life Big Data problem, a massive protein database of complex plant genomes of size 80 GB collected from the DSR, Indore under the ICAR for the categorization of protein sequences into the respective superfamilies. This categorization helps to assists DSR scientists to enhance the productivity of next generation protein sequences of different species of plant. The productivity of next generation protein sequences can be enhanced by making protein sequences more resistant to disease, drought, heat, high temperature, and food allergies. To do this, first, a rigorous study and analysis of protein sequences of complex plant genomes are carried out, and then we propose a novel feature extraction approach that extracts fixed-length numeric feature vectors for protein sequences for their effective categorization. Then, the performance of the scalable clustering algorithm is investigated on a massive protein database in terms of parameters assessing the quality of clustering results. Our approaches achieve a good trade-off between the quality of clustering results and the computational effort required to reduce run-time.

1.1 Motivation and Scope

This dissertation is a study of improving the performance of fuzzy clustering as well as the design and analysis of fuzzy based incremental and scalable clustering algorithms for handling VL and Big Data.

In the past few decades, cluster analysis has played a vital role in machine learning and pattern recognition. Recently, the use of fuzzy clustering algorithms is widely accepted for organizing unstructured data due to its capability to handle uncertainty. In spite of the wide acceptance of fuzzy clustering, it suffers from a problem of local optima due to a random selection of param-

eters such as the number of clusters, the location of cluster centers, and the fuzzifier. Therefore, the fuzzy clustering does not guarantee unique clustering results and thus the final clustering results may not be the optimal ones and lead the algorithm to converge into local optimal solutions [26]. There are many aspects of the design of a fuzzy clustering approach could be addressed, which may lead to an improvement in the performance of fuzzy clustering. Therefore, there is a need to explore the problem of selecting the optimal parameters of fuzzy clustering that can prevent the algorithm to converge into local optimal solutions.

Nowadays, Big Data has become prevalent because a vast amount of data collected every day from various sources such as sensor network, social networking sites, videos from the surveillance system, and genome projects. Mining valuable information in the prevalent Big Data is crucial to getting an edge on the competitive parties such as 30 billion pieces of content are shared on Facebook every month [45]; while photo-sharing sites such as Flickr, Instagram and Facebook are anecdotally known to possess 10s of billions of images, again taking up TBs of storage [41]. Hence, there is a need to design incremental and scalable clustering approaches for handling VL and Big Data problems. To design such a scalable clustering algorithm, big data analytics frameworks are required. Apache Spark is one of the most widely used Big Data analytics frameworks to make an iterative clustering algorithm scalable for handling Big Data. It is an open source cluster computing framework, developed to optimize large-scale interactive computation and works well for iterative algorithms by supporting in-memory computations. The proposed scalable algorithms are designed to handle Big Data by using the Apache Spark cluster computing framework. To work out the usefulness of the proposed scalable algorithms on a real-life problem of Big Data, we have collected and analyzed a huge complex plant genomes protein database and designed a novel approach for feature extraction to extract the numeric feature vectors of fixed-length from a protein database for enhancing its classification accuracy. Then to investigate the performance of the proposed scalable clustering algorithms, it is tested on a huge protein database for its efficient categorization to the respective superfamilies.

In this thesis, we investigate optimal parameters value assignment for FCM to improve its performance. Further, investigations are done to propose a novel

incremental clustering algorithm followed by a classification mechanism. The performance of the novel incremental clustering algorithm integrated with classification approach is investigated on different benchmark datasets and measured in terms of classification accuracy and run-time. In addition to this, novel scalable clustering algorithms are proposed to solve Big Data problems. Also, the novel scalable clustering algorithms are tested on various benchmark Big Datasets in terms of performance metrics related to handling Big Data. The analytical study is also performed for scalable clustering algorithms in terms of space and time complexity. Furthermore, the trade-off between performance gain versus accuracy is also assessed for in-depth analysis of experimental results. In addition to this, we have evaluated the performance of the proposed scalable algorithm on a real-life problem of Big Data. For this, we have collected a massive protein database of complex plant genomes from the DSR, Indore. Then, for the preprocessing of this large protein database, a novel feature extraction approach is proposed. The proposed feature extraction method represents each protein sequence with a fixed length numeric feature vectors which lead to an improvement in classification of protein sequences to the respective superfamilies. Then the proposed scalable clustering algorithm is tested on a massive protein database for the categorization of protein sequences into respective superfamilies.

1.2 Objectives

The research work is addressed to tackle the different issues in fuzzy clustering for improving its performance and development of an incremental clustering algorithm for handling VL data. Further, the focus of research is to develop scalable fuzzy clustering algorithms for efficient handling of Big Data problems. To demonstrate the applicability of proposed scalable fuzzy clustering algorithm on a real-life Big Data problem, the collection of complex plant genomes protein database is done from the DSR, Indore that comprises of protein sequences of different species of a plant. The objectives of our research work are defined as follows:

1. To find the optimal number of clusters for fuzzy clustering we propose a novel cluster validity index which does the following:

(a) Increase the intra-cluster compactness and inter-cluster separation and reduce the inter-cluster overlap between fuzzy clusters.
(b) In this, all the necessary measures of cluster analysis are incorporated to capture the underlying structure of data more accurately.

2. To find an optimal fuzzifier, we propose a QIE-FCM algorithm that has the following features:
(a) Quantum computing is utilized in fuzzy clustering, to provide a better characteristic of population diversity and to maintain a proper balance between exploration and exploitation.
(b) This enables us to efficiently explore a search space and exploit the optimal solution of fuzzifier.

3. To determine an optimal value of all the parameters of the fuzzy clustering algorithm, i.e., the initial cluster centers, the number of clusters, and the fuzzifier, we propose an EQIE-FCM algorithm that has the following characteristics:
(a) It utilizes quantum computing principle in fuzzy clustering process to appropriately explore a search space and to exploit the optimal solution of all the parameters of fuzzy clustering.
(b) Attaining the optimal solution of all the parameters that help to eliminate the local convergence problem.

4. To propose an incremental fuzzy based iterative clustering algorithm for addressing VL data, that utilizes the concept of Bayes' theorem to design it as a classifier for the effective classification of VL data.

5. To propose and develop, scalable fuzzy based iterative clustering algorithms for handling Big Data that speedup the clustering process of Big Data by reducing the run-time and optimizing the storage space. Also, to propose and develop an analytical formulation for space and time complexity.

6. To investigate the performance of the proposed variants of fuzzy clustering algorithms in comparison to other approaches on several benchmark datasets. The performance is evaluated in terms of optimal parameters of fuzzy clustering, accuracy, run-time, NMI [46], ARI [47, 48],

F-measure [45], objective function, speedup, sizeup, and scaleup [49, 50], and the trade-off between performance gain versus accuracy, space and time complexity.

7. To apply the proposed scalable clustering algorithms on a real-life Big Data problem, i.e., the categorization of protein sequences of the complex plant genomes.

8. To propose a novel algorithm for protein sequence database, which extracts numeric feature vectors of fixed-length from a large protein database by encoding each protein sequence and considering both the local and global similarity measure for improving the classification performance which is observed in terms of classification accuracy, sensitivity, specificity, and the confusion matrix.

9. To investigate the performance of the proposed scalable clustering algorithm on a large protein database for their categorization into the respective superfamilies is analyzed in terms of NMI [46], F-measure [45], and accuracy.

1.3 Contributions

The major contributions of the work done in this field are to enhance the understanding of the underlying structure of data by forming more appropriate clusters of data. This cluster formation is further analyzed in the larger scenario for handling VL and Big Data that helps to deal with data of any size. Thus giving a generalized means to learn the machine with any size of data. These contributions are divided into two broad categories. Firstly, the design of variants of fuzzy clustering algorithms for determining the optimal parameters of fuzzy clustering. Secondly, the design of an incremental clustering algorithm for handling VL data. Furthermore, the design of scalable fuzzy clustering algorithms to handle Big Data problems. The details of the same are given below.

1. To evaluate the optimal number of clusters, a novel cluster validity index is designed by exploiting three proposed measures defined as follows:
   (a) Intra-cluster Compactness,

(b) Inter-cluster Separation based on Fuzzy set,

(c) Inter-cluster Overlap.

It determines the optimal number of clusters in such a way that the data points present within the cluster are tightly coupled with each other, the separation between the data points present in different clusters are higher and it allows the minimum overlap between the data points present in different clusters.

2. To identify the optimal weighted exponent or fuzzifier from a large search space, a new hybrid model is proposed, referred to as the QIE-FCM. The proposed approach utilizes the merits of both the quantum computing and fuzzy concepts to improve the performance of fuzzy clustering by finding its optimal parameters.

3. This work is further enhanced to find the optimal value of all the parameters of fuzzy clustering, i.e., the weighted exponent, the number of clusters, and a set of initial cluster centers. An EQIE-FCM algorithm is proposed to efficiently utilize a large search space for the selection of optimal values of all these parameters.

4. To handle the VL data efficiently, we propose the RSIO-FCM algorithm. The proposed approach proceeds by partitioning the VL data into various subsets or chunks, and then sequentially performs clustering of each subset. The major advantage of this approach is that the clustering results of a previous subset are utilized for the clustering of the current subset. Thus, it ensures that the clustering of these subsets adequately covers the entire sample space represented by VL data. Furthermore, the concept of Bayes' theorem is utilized in the proposed clustering algorithm to design it as an effective classifier for handling VL data. Thus, this designed classifier improves the classification accuracy by minimizing the objective function and it also reduces the run-time. The proposed algorithm suffers from a limitation, that it produces highly deviated cluster centers because it might be possible that two subsets would consist of samples belonging to a distinct class. Due to this reason, if the cluster centers of the previous subset are fed as an input for the clustering of the current subset, then it

will converge slowly and produces highly deviated cluster centers. Hence, results in a higher number of iterations during clustering of the current subset.

5. To overcome the limitations of RSIO-FCM and to efficiently handle Big Data problems, a fast and scalable fuzzy based clustering algorithm is designed referred to as SRSIO-FCM. It processes the Big Data by partitioning it into various subsets and performs parallel processing of each subset using the Apache Spark Big Data processing framework. For this, we create a setup of Apache Spark Cluster with 3 nodes, where each node has the following configuration: Intel Xenon(r) CPU E5-1607 v3 @ 3.10GHz x 4 with 32GB RAM and 2TB storage. In the proposed approach, we eliminate the need for storing the membership matrix, which makes the execution of the proposed algorithm faster by reducing the run-time. In addition to this, we resolve the problem of using the highly deviated cluster centers for the clustering of the current subset. Thus, it provides faster convergence by taking fewer iterations during clustering of each subset. It also shows a significant improvement in run-time by improving the space and time complexity. The same is verified by experiments. For the purpose of fair comparison of SRSIO-FCM, we also designed, a scalable model of the existing LFCM algorithm and the rseFCM algorithm referred to as SLFCM and SrseFCM. The performance of the proposed SRSIO-FCM is evaluated in comparison with SLFCM and SrseFCM, respectively.

6. The proposed hybrid models have been tested on some standard benchmark datasets and compared with other state-of-the-art soft computing approaches. The hybrid models are designed to improve the efficacy of fuzzy clustering, and the performance of these hybrid models are evaluated in terms of an optimal value for the number of clusters, the fuzzifier parameter, the initial cluster centers, and the number of iterations. One more incremental clustering algorithm integrated with a classification mechanism is proposed and the performance is evaluated on various benchmark datasets in terms of classification accuracy and run-time. Also, the performance of scalable clustering algorithm is assessed in terms of NMI, ARI, F-measure, objective function, speedup, sizeup, scaleup, and the trade-off

between performance gain versus accuracy.

7. The proposed scalable fuzzy clustering algorithms are also tested on a real-life Big Data problem, i.e., categorization of protein sequences of a complex plant genome. For this, an exhaustive study and analysis are carried out on the protein database collected from the DSR, Indore. Then the preprocessing of protein sequences present in the protein database is performed to convert these protein sequences into numeric feature vectors. For this, we propose a novel feature extraction algorithm that represents a long chain of protein sequences with fixed-length numeric feature vectors. This algorithm performs encoding of protein sequences by considering both local and global similarity of occurrences of amino acids in protein sequences. The proposed feature extraction algorithm leads to the reduction of the actual database file, and improvements can be observed in terms of classification accuracy, sensitivity, specificity, and the confusion matrix.

8. On the preprocessed form of protein database, the performance of scalable clustering algorithms is investigated for the categorization of protein sequences of a complex plant genome into the respective superfamilies in terms of NMI, F-measure, and accuracy.

1.4 Organization of the Thesis

In this thesis, investigations are done for the selection of optimal parameters for fuzzy clustering for improving its performance and design the incremental and scalable fuzzy clustering algorithms for handling VL and Big Data problems. For the investigation of optimal parameters of fuzzy clustering, a novel cluster validity index is proposed by integrating intra-cluster compactness, inter-cluster separation, and inter-cluster overlap measures and also propose the hybrid quantum-inspired evolutionary fuzzy clustering algorithms. Furthermore, an incremental clustering algorithm is proposed for classification of VL data. In addition to this, scalable fuzzy based clustering algorithms are proposed for handling Big Data. To investigate the application of the proposed scalable clustering algorithm for solving a real-life Big Data problem (categorization of

protein sequences of a complex plant genome), a protein database of massive size is collected from the DSR, Indore. Then, this huge protein database is preprocessed by proposing a novel feature extraction approach. After this, we apply the proposed scalable clustering algorithm on this massive protein database. Thus, the overall work is divided into six logical sections which are organized as follows:

Chapter 2 details the background study of the previous work undertaken in the field of clustering. The chapter starts with the introduction of clustering algorithms where we describe the variations of fuzzy clustering and the important parameters affecting the performance of fuzzy clustering. After that, an overview of recent and existing cluster validity indexes and the hybrid approaches for selection of optimal parameters of fuzzy clustering are presented. A survey of accelerated and incremental fuzzy clustering algorithms designed for handling large and VL data are presented along with a discussion of existing Big Data processing frameworks. A study of some feature extraction approaches designed for protein sequences of a plant genome is also presented. Finally, the definitions and detailed description of performance measures are presented along with a brief overview of datasets used in the experimental study.

Chapter 3 addresses the problem of selecting the optimal number of clusters for fuzzy clustering and presents a novel cluster validity index. The proposed cluster validity index jointly incorporates the three measures, i.e., intra-cluster compactness, inter-cluster separation, and inter-cluster overlap. It captures the underlying structure of different datasets, and it is reliable in detecting the optimal number of clusters. The chapter finally reports the experimental evaluation, in which we compare our proposed cluster validity index with other cluster validity indexes from literature.

Chapter 4 presents the selection of optimal parameters of fuzzy clustering, i.e., fuzzifier, the number of clusters, and the initial cluster centers. In this chapter, we present the details of the proposed QIE-FCM algorithm. It identifies the optimal fuzzifier and number of clusters for fuzzy clustering from a large search space and maintains a proper balance between exploration and exploitation. In this chapter, experimental results evaluate the usefulness of the QIE-FCM algorithm for identifying an optimal value of the fuzzifier and the number of clusters correctly from a large search space. Furthermore, an enhanced version

of the QIE-FCM algorithm called EQIE-FCM is proposed for the selection of the optimal value of parameters of fuzzy clustering, i.e., the fuzzifier, the number of clusters, and the initial cluster centers from a large search space. The EQIE-FCM algorithm eliminates a local convergence problem which occurs in the QIE-FCM algorithm due to the random selection of initial cluster centers. This chapter finally presents the verification of performance for the QIE-FCM algorithm which shows its reliability in identifying the optimal value of all the parameters correctly from a large search space. Thus it achieves a significant reduction in the number of iterations to reach an optimal solution.

In chapter 5, we propose an incremental clustering algorithm for handling VL data, i.e., the RSIO-FCM algorithm. This algorithm utilizes the concept of Bayes' theorem to design a classifier for the effective classification of VL data. RSIO-FCM accelerates the speed of the clustering process by processing the entire data chunk by chunk. This approach captures the whole sample space by performing clustering on each chunk of data, and thus the clustering result of the previous chunk is used for the processing of current chunk. The proposed approach improves the classification accuracy and significantly reduces the run-time. The chapter finally reports the experimental evaluation, which compares the classification accuracy and run-time of proposed approach with other clustering algorithm.

In chapter 6, we proposed a novel clustering algorithm for handling Big Data called the SRSIO-FCM algorithm. It is implemented using a Big Data processing framework called Apache Spark and tested on Apache Spark cluster for handling Big data. This chapter presents the details on how the proposed approach scales for Big Data in comparison with other algorithms. The chapter finally, reports the experimental evaluation that compares our SRSIO-FCM algorithm with other proposed scalable models, i.e., the SLFCM and the SrseFCM.

In chapter 7, the novel scalable clustering algorithm is tested on a real-life complex plant genome (protein) data. First, the plant genome data are thoroughly analyzed. Then, for its preprocessing, we present a novel feature extraction method called the CPSF approach. The proposed approach represents a long chain of the protein sequence with a fixed-length numeric feature vector. The proposed feature extraction approach is designed to extract fixed-length features corresponding to protein sequences that lead to an improvement in

classification accuracy. The chapter finally presents the experimental results in which the performance of the proposed feature extraction approach is evaluated on various well-known classifiers and also compared with other feature extraction methods designed for protein sequence classification. The experimental results show an improvement in the classification accuracy in comparison with other approaches. In addition to this, the performance of the proposed scalable clustering algorithm is evaluated on the real-life massive protein database.

Finally, in chapter 8, conclusions are drawn regarding the research results of each of the addressed problems and the overall work in the thesis. The contributions to the state-of-the-art of the tackled subjects are outlined, and some future work directions are also highlighted.

Chapter 2

Literature Survey

This chapter discusses various clustering algorithms and issues related to those clustering algorithms. Then, the focus of the discussion shifts to the FCM clustering algorithm and matters related to its design. After that, we give a detailed description of parameters related to FCM, i.e., the number of clusters (c), the fuzzifier (m), and choosing the set of initial cluster centers (V) that significantly affects its performance. Further, a survey of clustering algorithms for processing VL datasets is carried out. Then, the study related to Big Data processing frameworks is presented along with the parallel processing algorithms for handling Big Data. A real-life Big Data problem, i.e., classification of protein sequences of various plant genomes is analyzed, and a survey related to its feature extraction approaches are presented. For the experimental evaluation of the proposed algorithms, performance measures and datasets for testing are presented.

2.1 Basic Clustering Algorithms

Clustering is one of the widely used data analysis techniques used to discover groups and underlying structure in a set of data samples. The aim of clustering is to form a grouping such that the data points belonging to the same group are more similar to each other than those which are lying in different groups [51,52]. Clustering has been receiving tremendous attention in many application domains. These areas include gene analysis [53], indexing and compression [54], image analysis [55], signal processing [56], text classification [57, 58], cyber security, and data mining [59]. Over the last five decades, many cluster-

ing algorithms [52, 60–62] have been developed based on various theories and applications. A taxonomy of clustering algorithms was described by Fahad et al. [62]. An overview of this taxonomy of clustering algorithms is presented in Figure 2.1. In the taxonomy of clustering algorithms, many distinctions of clustering algorithms are presented, but our focus is on the distinction of hierarchical and partitional clustering algorithms [1, 16, 17]. Hierarchical clustering methods yield an entire hierarchy, i.e., a nested sequence of partitions of the input data. At each level of the hierarchy, there is a different number of clusters. A partitional clustering algorithm seeks to obtain a single partition of the input data into a fixed number of clusters, usually by optimizing an objective function. Partitional methods produce clusters by optimizing a criterion function, defined either locally (on a subset of the patterns) or globally (defined over all of the patterns). Partitional methods are advantageous in applications involving large datasets for which construction of a dendrogram is computationally prohibitive. Partitional clustering is broadly divided into hard (or crisp) and fuzzy methods. The details of both these categories of the algorithms are discussed below.

2.1.1 Hard Clustering

In hard clustering algorithms, each sample of the dataset must be assigned to precisely one cluster. Hard clustering methods are based on classical set theory and require that a sample either does or does not belong to a cluster. Hence, the clusters in a hard partition are disjoint. A major drawback of hard clustering

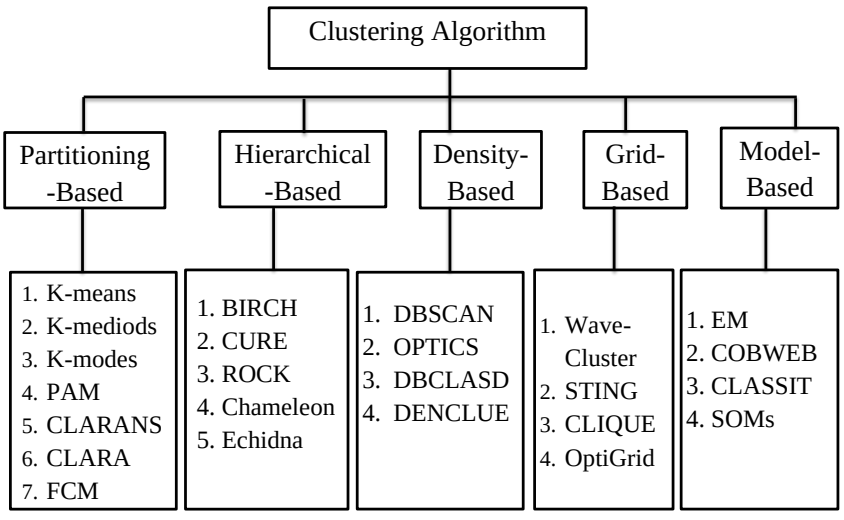


Figure 2.1: A sample taxonomy of clustering algorithms.

techniques is that they may lose some valuable information, such as they are unable to capture the structure of real datasets in which there are no definite boundaries between the clusters. Due to the loss of such vital information, sometimes hard clustering leads to a meaningless grouping [21].

K-means clustering

The simplest algorithm of this type is the K-means algorithm with squared error criterion [63]. However, there exist multiple variants of this algorithm [60, 61]. K-means is a distance-based partitioning algorithm [1]. The main idea is to define a set of c clusters. The next step is to compute the distance of each data point to the cluster and associate each data point to the nearest cluster. The aim is to reduce the sum of squared error, represented by the square of the Euclidean distance between each data point and its closest respective cluster center [1, 18, 61]. The value of the sum of squared error for a partition is given as follows:

$$J(X, V) = \sum_{j=1}^{c} \sum_{i=1}^{n} ||x_i - v_j||^2 \tag{2.1}$$

where J is the sum of the squared error, X is a dataset consisting of n data points, and each data point consists of d dimensions, such that $X = \{x_1, x_2, .., x_n\} \in \mathcal{R}^{d \times n}$. The initial cluster centers are represented by a set $V = \{v_1, v_2, ..., v_c\} \in \mathcal{R}^{d \times c}$ such that a set V consists of c clusters and cluster center of c^{th} cluster is denoted by v_c which consists of d dimensions, and $||x_i - v_j||^2$ is the euclidean distance measure between a i^{th} data point x_i and the j^{th} cluster center v_j. The objective function J is an indicator of the distance of the n data points from c cluster centers. Finding the set of cluster centers that minimizes J is an NP-hard problem [64]. Here, an initial set of cluster centers is required for the computation of Eq. (2.1). There exist several initialization strategies [1], the most common approach is to randomly select few data points to provide the initial positions of the cluster centers [18]. The K-means clustering terminates if the difference between successive values of J does not exceed a user-defined value. ***Algorithm 2.1*** is composed of the following steps, defined as follows [1, 18]:

Algorithm 2.1 Algorithm for K-means Clustering

1: **Begin**
2: **Place** c points into the space represented by the data points that are being clustered. These points represent a set of initial cluster centers V.
3: **Assign** each data point to the nearest cluster centers.
4: **When** all data points have been assigned, recalculate the positions of the cluster centers using Eq. (2.2) such that X_j is the subset of data points from X belonging to the j^{th} cluster.

$$v_j = \frac{1}{|X_j|} \sum_{x_i \in X_j} x_i \tag{2.2}$$

5: **Repeat** steps 2 and 3 until there is a relatively small change in the cluster centers. The produced partition minimizes the objective function.
6: **End**

There are many limitations of the K-means clustering. The first is that the K-means algorithm requires an initial set of cluster centers, which implies that the number of clusters is to be known in advance [1]. The second is that the K-means algorithm is non-deterministic if this initial set of clusters is chosen randomly [65]. The final set of cluster centers returned by K-means is highly dependent on the initial set of clusters centers. Third, the K-means clustering is highly sensitive to outliers, although it is quite efficient in terms of computational time.

Run-time complexity

The K-means clustering algorithm has a time complexity of $O(ntdc)$, where n is the number of data points, t is the number of iterations, d is the number of dimensions of data points or features, and c is the number of clusters [18].

2.1.2 Fuzzy Clustering

Fuzzy clustering is based on a fuzzy set theory which allows a data point to have varying grades of membership in a set [66]. Fuzzy set theory is used in the clustering algorithm so that a data point can belong to multiple clusters [67]. A fuzzy clustering produces fuzzy partitions that can be expressed by the membership matrix or fuzzy partition matrix, U of size $n \times c$. The degree of membership of data point i in cluster j is represented by u_{ij}. This is subjected

to the following constraints [22, 68]:

$$u_{ij} \in [0,1], 1 \leq i \leq n, 1 \leq j \leq c \tag{2.3}$$

$$\sum_{j=1}^{c} u_{ij} = 1, 1 \leq i \leq n \tag{2.4}$$

$$\sum_{i=1}^{n} u_{ij} > 0, 1 \leq j \leq c \tag{2.5}$$

Fuzzy cluster analysis gives the flexibility to express that the data points can belong to more than one cluster at the same time. In addition to this, the membership degrees can also show how ambiguously or definitely a data point should belong to a cluster. The concept of fuzzy analysis has been successfully integrated into many clustering algorithms [69–72]. One of the most widely used fuzzy clustering algorithms is discussed next.

Fuzzy C-Means algorithm

The FCM algorithm was initially presented by Dunn [73] and completed by Bezdek [22]. It is also known as LFCM clustering [36]. LFCM partitions the collection of n data points x_i, $i = 1, ..., n$ into c fuzzy clusters and finds a set of cluster centers. LFCM executes iteratively until the difference in the cluster centers of previous and current iteration is less than the defined termination criteria (ϵ) [68]. The aim is to minimize the objective function (J_m). In LFCM, the membership degree u_{ij} can take any values between 0 and 1. The objective function of LFCM can be formulated as follows:

$$J_m(U, V^{'}) = \sum_{i=1}^{n} \sum_{j=1}^{c} (u_{ij})^m \|x_i - v_j^{'}\|^2 \tag{2.6}$$

where $V^{'}$ denote the set of final cluster centers such that $V^{'} = \{v_1^{'}, v_2^{'}, .., v_c^{'}\}$. The parameter m controls the degree of fuzziness of every data point between the fuzzy clusters and reflects the partition percentages for every point in each cluster, and it drastically affects the clustering results. The LFCM algorithm is based on the iterative optimization of the objective function given in Eq. (2.6) by updating the membership matrix U and cluster centers to get a set of final cluster centers $V^{'}$. The parameters related to LFCM are listed in Table 2.1.

Algorithm 2.2, presents the steps of the LFCM algorithm.

Algorithm 2.2 Algorithm for Literal Fuzzy C-Means (LFCM) to Iteratively Minimize $J_m(U, V^{'})$

1: **Input:** X, V, c, m, ϵ
2: **Output:** $U, V^{'}$
3: **Begin**
4: **Randomly** initialize cluster centers $V = \{v_1, v_2.., v_c\}$.
5: **Compute** the cluster membership matrix U using Eq. (2.7) such that $\sum_{j=1}^{c} u_{ij} = 1$.

$$u_{ij} = \frac{\parallel x_i - v_j \parallel^{\frac{-2}{m-1}}}{\sum_{k=1}^{c} \parallel x_i - v_k \parallel^{\frac{-2}{m-1}}}, \forall i, j \tag{2.7}$$

6: **Compute** the cluster centers $v_j^{'}$ for $j = 1, 2, .., c$.

$$v_j^{'} = \frac{\sum_{i=1}^{n} [u_{ij}]^m x_i}{\sum_{i=1}^{n} [u_{ij}]^m}, \forall j \in [1, c] \tag{2.8}$$

7: **If improvement** in $J_m(U, V^{'})$ is less than ϵ or **if** $\parallel V^{'} - V \parallel < \epsilon$, then stop; **Else**: $V = V'$ and go to Line 5.
8: **Return** $U, V^{'}$
9: **End**

The LFCM algorithm has an expected run-time complexity of $O(ntdc^2)$ [33]. Despite the wide acceptance of the LFCM algorithm, it suffers from several disadvantages. There are many parameters in LFCM, which significantly affects the performance and results of fuzzy clustering. An important issue in LFCM is the selection of the number of clusters and choosing the initial cluster centers. These parameters are selected randomly, and it makes the iterative process converge to a locally optimal solution [77]. Many methods have been proposed by researchers [27, 78–80], to automatically determine the number of clusters and locations of cluster centers. However, these methods do not provide good generalization capabilities in obtaining appropriate centers [78].

Table 2.1: Specification of parameters for LFCM.

Parameters	Description	Values
ϵ	Termination criteria	0.001 [22]
m	Fuzzifier	$m \in [1.5, 2.5]$ [74]
$c_{\min}$	Minimum number of clusters	2
$c_{\max}$	Maximum number of clusters	$\sqrt{n}$ [4, 74–76]
n	Number of data points	Size of dataset
c	Number of clusters	$c \in [c_{\min}, ..., c_{\max}]$ [8]

Another important factor that influences the effectiveness of LFCM is the fuzzifier [24, 25]. The fuzzifier controls the degree of fuzziness of a data point between the fuzzy clusters. Most researchers have shown that there exists an efficient range of values for the fuzzifier. However, the methods adopted by researchers are mainly experimental or empirical. Hence, selecting an appropriate value for the fuzzifier while implementing the LFCM is still an open problem. Another major difficulty with applying the LFCM clustering algorithm is scalability. On large datasets, defined as 10^8 bytes according to Huber's description [28], the time taken by traditional clustering algorithms is too long. Therefore researchers have prohibited the use of conventional algorithms and have designed new accelerated clustering algorithms [32–34, 36, 81] that perform clustering by processing the data in terms of subsets [81].

Thus, a wide range of issues affecting the performance of LFCM during the clustering process, like the identification of an appropriate number of clusters, the selection of the fuzzifier, and the scalability measured in terms of speedup and sizeup for processing VL and Big Data. To handle these issues, it is proposed to investigate the improvements in the performance of LFCM and to design the incremental along with scalable fuzzy approaches for efficient clustering of VL and Big Data. Related to these issues of fuzzy clustering, a lot of work is carried out [24, 42, 74, 82]. One of the important issues related to fuzzy clustering based algorithms is discussed, which is termed as a cluster validity index. The cluster validity index helps in identifying the number of clusters, and it also accesses the quality of formed clusters.

2.2 Preliminaries for Cluster Validity Index

In clustering scenario, there are two fundamental questions [83] that need to be addressed: (i) how many clusters are present in the data and (ii) how real or useful is the clustering in itself. In fuzzy clustering, finding the number of clusters has been one of the most severe problems. Many methods have been proposed by the researchers [27, 78–80] that automatically determines the number of clusters. However, these methods do not provide good generalization capabilities in determining the number of clusters and obtaining appropriate cluster centers. As these methods unable to determine the number of clusters

appropriately for a different variety of datasets, therefore these methods cast into the problem of model selection [52]. One approach uses the principle of minimum description length for selecting the number of clusters [84]. Gap Statistics [85] is another commonly used method for deciding the number of clusters. In spite of these objective criteria, it is not easy to determine the number of clusters that leads to more meaningful clusters.

The problem of determining a proper number of clusters and the corresponding fuzzy partition is an important issue in cluster analysis, which is usually evaluated by a cluster validity index. A cluster validity index is an evaluation function that measures, relatively, the quality of c discovered clusters in each fuzzy partition. Therefore, the best partition can be decided as the one with the largest and smallest value of cluster validity index among all the fuzzy partitions [5, 86–88]. Many approaches are based on this principle and finds the optimal number of clusters by repeatedly running the algorithm with a different number of clusters as an input. Then, select the partitioning of the data as the best partition that results in the largest or smallest value of the cluster validity index among all the fuzzy partitions [89]. Ideally, a validity index should take care of the following aspects of partitioning [5, 90]:

- **Compactness measure**: It evaluates the concentration of data points that belong to the same cluster. A standard measure of compactness is the variance, which should be minimized. Thus, the minimum value of variance of patterns in a cluster indicates cluster compactness [5].

- **Separation measure**: This measure quantifies the distance between fuzzy clusters where a large distance represents a greater separation.

- **Overlap measure**: It quantifies the degree of overlap of a specified number of fuzzy clusters.

To evaluate the quality of clustering results, many cluster validity indexes [76, 91–98] have been proposed in literature such as Partition Coefficient (V_{PC}) and Partition Entropy (V_{PE}) indexes [91], Fukuyama and Sugeno (V_{FS}) index [92], Xie and Beni (V_{XB}) index [76], Rezaee et al. (V_{CWB}) index [93], CS (CSI) index [94], Kim and Lee (v_{OS}) index [95], PBMF index [96], PCAES index [97], and OSI index [98]. These validity indexes incorporate only two measures out

of three measures discussed above. The details of some of these validity indexes are presented next.

In particular, Bezdek proposed two commonly used cluster validity indexes [91] for fuzzy clustering, the Partition Coefficient (V_{PC}) and the Partition Entropy (V_{PE}) which is defined as:

$$V_{PC} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^2 \tag{2.9}$$

$$V_{PE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}\log_2(u_{ij}) \tag{2.10}$$

The Partition Coefficient measures the amount of overlap between clusters, whereas Partition Entropy only measures the fuzziness of a cluster partition. A large value of V_{PC} and a small value of V_{PE} index implies better clustering results. The major drawback of these indexes is that as the number of clusters increases, the V_{PC} index monotonically decreases while the V_{PE} index value monotonically increases, which makes these indexes hard to detect the optimal number of clusters. Also, these indexes have no direct relationship with the data it means that they use only the fuzzy membership degree for each cluster without considering the data structure of the clusters [95].

Fukuyama and Sugeno proposed a cluster validity index (V_{FS}) [92] that combines the properties of compactness and separation measurements. The minimum value of V_{FS} index indicates the optimal fuzzy partition. The V_{FS} index is defined as follows:

$$V_{FS} = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^m \|x_i - v_j\|^2 - \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^m \|v_j - \bar{v}\|^2 \tag{2.11}$$

where $\bar{v}$ denotes the mean of the cluster centers.

Xie and Beni proposed a validity index (V_{XB}) [76] that evaluates overall compactness of fuzzy c-partition according to geometric distance measure and separation between the cluster centers. This index monotonically decreases when the number of clusters increases. The minimum value of V_{XB} index indicates the optimal fuzzy partition. The V_{XB} index is defined as:

$$V_{XB} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^2 \|x_i - v_j\|^2}{n(\min_{j\neq k} ||v_j - v_k||^2)} \tag{2.12}$$

Rezaee et al. proposed a new validity index (V_{CWB}) [93] that tends to focus on compactness and separation measures. In this index, the function describing the compactness measure is combined with the separation measure. Here, the compactness measure is computed within the fuzzy cluster based on the degree of variance, which indicates the average scattering of c clusters. In this, a separation measure is calculated by considering the distance between the fuzzy clusters. The minimum value of this index indicates the optimal number of clusters and fuzzy partitions. The V_{CWB} index is defined as:

$$V_{CWB} = \tau \frac{\sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}(x_i - v_j)^2}{cn\|\sigma(X)\|} + \frac{D_{\max}}{D_{\min}} \sum_{k=1}^{c} \left(\sum_{z=1}^{c} \|v_k - v_z\| \right)^{-1} \tag{2.13}$$

where $\sigma(X)$ represents the variance of the pattern set X, τ accounts for a balancing factor, and $D_{\max}$ and $D_{\min}$ are the maximum and minimum distances between the cluster centers.

Kim and Lee proposed a new cluster validity index by considering the overlap and separation measures denoted by v_{OS} [95]. The minimum value of numerator indicates the classification of each data point x_i to a particular cluster from a given fuzzy c-partitions. The denominator computes the separation between clusters using a distance measure in the fuzzy set theory, which is based on the similarity measure [99]. The higher value of the denominator indicates well separated fuzzy partitions. The minimum value of v_{OS} index indicates the optimal fuzzy partitions. The v_{OS} index is defined as follows:

$$v_{OS} = \frac{\frac{2}{c(c-1)} \sum_{z=1}^{c-1} \sum_{y=y+1}^{c} \times \left[\sum_{u} \sum_{i=1}^{n} \delta(x_i, u : F_z, F_y) \omega(x_i) \right]}{1 - \underbrace{\min}_{z \neq y} \left[\underbrace{\max}_{x \in X} \min(u_{F_z}(x), u_{F_y}(x)) \right]} \tag{2.14}$$

where $\delta(x_i, u : F_z, F_y)$ determines whether two fuzzy clusters F_z and F_y are overlapped at the membership degree u for data point x_i, $\omega(x_i)$ represents weight factor for each data, $u_{F_z}(x)$ and $u_{F_y}(x)$ are the membership degree of each data point corresponding to two fuzzy clusters F_z and F_y.

Capitaine and Frelicot proposed a new cluster validity index OSI [98], which employs a multiple cluster overlap and a separation measure for each data point, both based on an aggregation operation of membership degrees. The OSI index

is defined as follows:

$$OSI = \frac{1}{n}\sum_{i=1}^{n}\frac{\perp_{l=2,c}^{1}(\perp_{j=1,c}^{c}u_{ij})}{\perp^{1}\underbrace{(\perp_{j=1,c}^{1}u_{ij},...,\perp_{j=1,c}^{1}u_{ij})}_{c-1\ \ times}} \tag{2.15}$$

where, $\perp_{j=1,c}^{l}u_{ij}$ is defined as the l-order fuzzy-OR operator [100] on $\{u_{ij}|j = 1, 2, ..., c\}$.

The above-discussed cluster validity indexes do not jointly incorporate all the measures necessary for cluster analysis, i.e., compactness, separation, and overlap. Some cluster validity indexes consider the compactness and separation measures to evaluate the optimal fuzzy partition while the other consider the separation and overlap measures. Thus, these indexes may not accurately evaluate the clustering results and may provide misleading information to the data analyst. Therefore, there is an extensive need for a cluster validity index, which jointly incorporates all the three measures for evaluating the optimal number of clusters and fuzzy partitions.

2.3 About Fuzzifier Parameter

Another important parameter for fuzzy clustering is the fuzzifier denoted by m. It is also called as weighting exponent and has a significant impact on the performance of FCM. FCM is sensitive to the initialization of m and usually gets trapped in local optima due to improper selection of m. When m is selected close to one, the FCM approaches the hard K-means algorithm and if m approach infinity, then the only solution for the FCM is the mass center of the dataset. Hence, choosing a suitable weighting exponent is a very challenging task when implementing FCM. However, different methods have been adopted by the researchers to select an appropriate value of the fuzzifier for the execution of FCM.

Pal and Bezdek [74] have given heuristic guideline regarding best choice of m, and they consider different cluster validity index to analyze the best selection of the fuzzifier. By experimental evaluation, the researchers found that suitable value for m is selected in the range of $[1.5, 2.5]$. Other researchers [101, 102] have given the similar recommendations, but these recommendations are based

on empirical studies and may not be appropriate in general for all the datasets.

Yu et. al [82] developed a new theoretical and practical approach for selecting the weighting exponent in FCM. In this method, a new local optimality test of solutions for the FCM is proposed. Based on this test, one obtains theoretical rules for selecting the weighting exponent in FCM, which show that a proper m depends on the dataset itself. However, theoretical analysis and numerical experimental results show that $m = 2$ is not a reasonable heuristic guideline.

Huang et al. [24], presented a theoretical approach consisting of two rules to determine the range of values for fuzzifier in the conventional FCM algorithm. This method utilizes the behavior of membership function on two particular data points, based on which it reveals the partial relationship between the fuzzifier and structure of the dataset. The results show that range of values of m identified is very close to range identified by other researchers from empirical studies.

From the sound analysis of FCM, Wu [25] proposed a new guideline for selecting parameter m. This guidance suggested that a large value of m makes FCM more robust to noise and outliers. However, considerably large m values that are greater than the theoretical upper bound would make the sample mean a unique optimizer. To avoid such unexpected cases, a simple and efficient method for fuzzy clustering is to assign a cluster core to each cluster. It means that if data points lie inside the cluster, then it would have membership values of zero or one. For a large theoretical upper bound case, the implementation of FCM with a suitably large m value is recommended. Otherwise, the implementation of FCM with cluster cores is suggested. In the case of large theoretical upper bound, when dataset contains noise and outliers, the fuzzifier $m = 4$ is recommended for FCM.

2.4 Deciding Other Parameters of Fuzzy Clustering Optimally

In FCM, there are few more parameters such as the number of clusters and the initialization of cluster centers in addition to the fuzzifier, which significantly affects clustering results. Determining the number of clusters has a significant role in obtaining practical and sound results for FCM. Similarly, choosing the

initial cluster centers is crucial as it has a direct impact on the formation of final clusters. The major problem with FCM is that it is sensitive to initialization of these parameters. The random selection of these parameters does not guarantee unique clustering results which make FCM algorithm to converge at local optimal solutions.

For the selection of the number of clusters and the fuzzifier, different validity indexes have been developed based on an unsupervised feature of the conventional FCM. Some of these validity indexes are V_{PC} and V_{PE} [91], V_{FS} [92], V_{XB} [76], V_{CWB} [93], CSI [94], v_{OS} [95], PBMF [96], PCAES [97], and OSI [98]. These validity indexes do not eliminate the local convergence problem of FCM. In addition to this, many hybrid fuzzy clustering approaches [27, 42, 43, 103, 104] have also been proposed by researchers to deal with the local convergence problem and for finding optimal parameters of FCM.

Hung et al. [43] proposed a QM-FCM approach for solving the local convergence problem of FCM. This method utilizes a large search space for initialization of parameters of FCM. Wang et al. [104] proposed an approach called RQECA. This method chooses a multi-distribution center location from traditional FCM. It is a combination of the RQIEA and FCM approach, to overcome the local search defect of FCM and make optimization result independent of a choice of initial values.

Kannan et al. [103] proposed a method called QEFCM to produce an alternative generalization of FCM clustering techniques to deal with more complicated data. This approach deals with efficient quadratic entropy FCM using the combination of regularization function, quadratic terms, mean distance functions, and kernel distance functions. It presents a complete framework of quadratic entropy approaching for constructing efficient quadratic entropy based fuzzy clustering algorithms. The proposed approach establishes an effective way of estimating membership degrees and updating centers by minimizing the objective functions. In this method, an efficient way of prototype initialization method is presented that assigns initial cluster centers without human intervention and also reduces the number of iterations of proposed approach. The Silhouette method [105] is used in this process to validate the fuzzy partitions and for choosing the number of clusters.

Bandyopadhyay and Maulik [42] proposed an approach that employs the

concept of VGA in fuzzy clustering to develop a novel nonparametric clustering technique. This method aims to evolve a proper value of the number of clusters by utilizing search capability of VGA, but it does not assume any particular underlying distribution of the dataset. Gan et al. [106] proposed the genetic fuzzy K-modes algorithm for clustering of categorical datasets. In this algorithm, they treated the fuzzy K-modes clustering as an optimization problem and used genetic algorithms to eliminate the local convergence problem of FCM. To speedup the convergence of algorithm, one-step fuzzy K-modes algorithm is integrated into the crossover process instead of traditional crossover operator.

Izakian and Abraham [27] proposed a hybrid fuzzy clustering method called as FCM-FPSO by integrating FCM algorithm with FPSO algorithm. In this approach, the concept of FCM is applied to improve the fitness of each particle and to make use of the merits of both algorithms to eliminate the local convergence problem of FCM. In literature, it is mentioned that the hybrid clustering algorithms have shown improvement in accuracy over traditional FCM but results in a bad execution time in comparison with partitional clustering techniques. Another problem with hybrid PSO based clustering algorithms is that PSO based approaches require tuning of an extensive range of parameters to find real solutions.

To overcome the problem of above discussed evolutionary approaches, we have used the principle of quantum computing. However, this principle has been taken from quantum physics, but in computer science, the research of evolutionary based quantum computing has been started since the late 1990s [107]. The research of evolutionary based quantum computing has been classified into two fields. One concentrates on generating new quantum algorithms using automatic programming techniques such as genetic programming [108]. The other concentrates on quantum-inspired evolutionary computing for a classical computer, a branch of study in evolutionary computing that is characterized by certain principles of quantum mechanics such as standing waves, interference, and coherence.

Here, quantum-inspired evolutionary computing principle is used to propose hybrid quantum-inspired evolutionary fuzzy clustering algorithms for selection of optimal parameters of fuzzy clustering. Preliminaries for the quantum computing theory is discussed next.

Quantum computing

The quantum computing represents data in terms of a quantum bit (Q). A quantum bit consists of several qubits represented as follows:

$$Q = (q_1|q_2.....|q_h) \tag{2.16}$$

where q_k represents the k^{th} qubit, $k = 1, 2, ...h$, and h represents the number of qubits to form a quantum bit (Q). Generally, qubits differ from classical bits in terms of representation. Classical bits represent only two possibilities of any event at one time by bit "1" or "0". However, a qubit can exist in both the states, i.e., "1" state and "0" state simultaneously using the probability concept proposed by Han and Kim [107, 109]. A characteristic of qubit representation is the ability to represent a linear superposition of "1" and "0" states probabilistically, which is denoted as follows:

$$q_k = \alpha_k \mid 0\rangle + \beta_k \mid 1\rangle, \tag{2.17}$$

where $0 \leq \alpha_k \leq 1$, $0 \leq \beta_k \leq 1$, α_k and β_k are the complex numbers representing the probability of a k^{th} qubit, and $k = 1, 2, ..., h$. According to Eq. (2.17), a qubit may appear in the "1" state, in the "0" state, or in a linear superposition of the two states, thus Eq. (2.16) can be rewritten as follows:

$$Q = \left\langle \begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \dots & \alpha_h \\ \beta_1 & \beta_2 & \dots & \beta_h \end{array} \right\rangle \tag{2.18}$$

where $(\alpha_k)^2 + (\beta_k)^2 = 1$ and $k = 1, 2, ..., h$. $(\alpha_k)^2$ is the probability that a k^{th} qubit found in state "1" and $(\beta_k)^2$ is the probability that a k^{th} qubit found in state "0". Since $(\alpha_k)^2 + (\beta_k)^2 = 1$, therefore Eq.(2.18) can be simplified as:

$$Q = \left\langle \begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \dots & \alpha_h \end{array} \right\rangle \tag{2.19}$$

As mentioned above, the advantage of qubit representation is that it can represent a linear superposition of two states, i.e., 0 state and 1 state probabilistically. For instance, an example is considered here that explains the essence of a quan-

tum bit formed using two-qubits is represented as:

$$Q = \left\langle \alpha_1 \middle| \alpha_2 \right\rangle \tag{2.20}$$

As discussed above, the value of α_k and β_k lies in the range of "0" and "1". Thus, α_1 and α_2 in Eq. (2.20) can be initialized with any value in the interval mentioned above as follows:

$$Q = \left\langle 1/\sqrt{2} \middle| 1/\sqrt{2} \right\rangle \tag{2.21}$$

As mentioned above that $(\alpha_k)^2 + (\beta_k)^2 = 1$, therefore with the given value of α_1 and α_2 in Eq. (2.21), the value of β_1 and β_2 is computed as $\beta_k = \sqrt{1 - (\alpha_k)^2}$. In Eq. (2.21) we can see that a quantum bit (Q) formulated in terms of two-qubits consist of 4 different stages are represented as follows:

$$\begin{aligned} Q = (1/\sqrt{2} \times 1/\sqrt{2})\langle 00\rangle + 1/\sqrt{2} \times 1/\sqrt{2})\langle 01\rangle \\ + 1/\sqrt{2} \times 1/\sqrt{2})\langle 10\rangle + 1/\sqrt{2} \times 1/\sqrt{2})\langle 11\rangle, \end{aligned} \tag{2.22}$$

In Eq. (2.22), we can see that a quantum bit (Q) with two-qubits would perform the operation on four values and thus it is enough to represent four states at the same time. Similarly, a quantum bit (Q) formed by h-qubits can represent 2^h states at the same time. Thus, quantum bit representation has better characteristics of population diversity (exploration) than other representations, since it can represent a linear superposition of states probabilistically. This exploration is achieved through the observation process [110].

In the observation process [110], a quantum bit (Q) with h-qubits, generate a random number vector $\rho = [\rho_1 \rho_2 ... \rho_h]$, where $0 \leq \rho_k \leq 1$, $k = 1, 2, ..., h$; and the corresponding bit in a binary vector $b = [b_1 b_2 .... b_h]$ takes "1" if $\rho_k \leq (\alpha_k)^2$, or "0" otherwise. This concept of qubit representation and observation process is discussed in detail in chapter 4, and it is utilized in the proposed hybrid quantum-inspired evolutionary fuzzy clustering algorithms for the selection of optimal parameters of fuzzy clustering from a large search space.

In recent years, it has been observed that traditional fuzzy based clustering algorithm is unable to perform clustering of large and VL datasets. In such

a case, there is a need to develop algorithms which can reduce run-time while maintaining the quality of clustering results. The detailed description of issues involved in the processing of large or VL data with fuzzy clustering algorithms and their related work is described next.

2.5 Clustering Algorithms for Large and Very Large Data

Clustering is a widely used unsupervised learning technique for exploratory data analysis. One of the major problems with traditional fuzzy clustering algorithms is that they do not work well for large datasets. As discussed earlier in section 2.1.2, LFCM has an expected run-time complexity of $O(ntdc^2)$. Kolen and Hutcheson [111] suggested that it is possible to reduce the run-time of LFCM to $O(ntdc)$.

Given a dataset with c natural clusters, LFCM can be accelerated further by reducing n, d or t. There are techniques available [112, 113] for reducing the number of dimensions of data, but many of these techniques are not integrated into the clustering process instead they only preprocess the data [114, 115]. An alternative approach called subspace clustering, find clusters using a subset of the available features [116, 117]. Each cluster formed in this way can use a different subset of available features. Thus, there is a need to design an accelerated approach that reduces the size of data and the number of iterations. One of the ways to decrease the number of iterations is to select the position of initial cluster centers in such a way that it is very close to the final solution. Due to the reduction in iterations, lesser run-time is achieved.

Methods are investigated to improve the initial starting points for LFCM by processing a small data sample. Cheng [30] describes an iterative process to develop a "good" starting point called mrFCM approach which consists of two parts. The first part progressively samples the dataset by improving the starting clusters until a termination criterion is satisfied. Then it uses these starting clusters to initialize LFCM on full dataset. Hung and Yang [31] has proposed an approach called psFCM, that partitions the data using a K-d tree to obtain a simplified dataset. It uses a subsample to estimate the position of

the cluster centers, and then the resulting estimate is used to initialize LFCM on the full dataset.

There are methods proposed by researchers [32, 34] that apply the extended clustering schemes on a clustering algorithm. These methods use a representative sample of the full dataset, and then it noniteratively extends the sample result to obtain clusters for the remaining data in the entire dataset. The algorithms which include their noniterative mechanisms for extension are referred to as extensible algorithms [32]. However, some algorithms are designed based on the sampling methods are CLARA [61], corsets [118], and CURE [119]. These algorithms work by randomly selecting a sample of data from a huge dataset and computing cluster centers of this sampled data. The results obtained on this sample are extended to approximate the cluster centers of the entire dataset. Pal et al. [32] and Wang et al. [34] used progressive sampling to select a subsample representative of the dataset. They used a divergence test to assess whether the subsample matched the distribution of the dataset. If the test failed, progressively larger subsamples were taken until the test passed.

One of the most well-known methods for fuzzy clustering of VL data is the geFFCM algorithm [29]. This algorithm uses statistics-based progressive sampling to produce a reduced dataset. However, it assumes that the reduced dataset is large enough to capture the overall nature of the data. It then clusters this reduced dataset and noniteratively extends the partition to the full dataset. However, the sampling method used in geFFCM can be inefficient in some cases, when the data reduction is not sufficient for VL data. One of the most simple ways to reduce the size of data is to select a sample of the dataset and apply the clustering algorithm to the sample. Havens et al. [36] adapt geFFCM algorithm into a rseFCM algorithm. This algorithm performs clustering over a sample of the VL data using LFCM [36,37], and then it extends results over the entire data using the steps of the LFCM algorithm. The details of the rseFCM algorithm are presented subsequently.

Furthermore, incremental fuzzy clustering algorithms have been developed based on the well- known LFCM [22]. Few traditional incremental algorithms include the spFCM [33] and the oFCM [38] algorithms. These two algorithms process data chunk by chunk and estimate the c centroids for the entire dataset by extracting information in each chunk. The spFCM and oFCM algorithms

both are based on a vector representation of data and referred to as object data.

Random sampling plus extension Fuzzy C-Means (rseFCM)

To address the clustering of VL data according to Huber statistics [28] presented in Table 1.1, Havens et al. [36] presented an algorithm. In this algorithm, he addressed a way to process VL data, is to sample the dataset and then LFCM is applied to the sample data to compute the cluster centers. ***Algorithm 2.3*** outlines the steps of the rseFCM algorithm.

Algorithm 2.3 Algorithm for random sampling plus extension Fuzzy C-Means to approximately minimize $J_m(U, V^{'})$

1: **Input**: X, V, c, m, ϵ
2: **Output**: $U, V^{'}$
3: **Sample** the n_s objects from X without replacement, denoted X_s.
4: $U_s, V = LFCM(X_s, V, c, m)$
5: **Extend** the partition (U_s, V) to X,$\forall x_i \notin X_s$ using Eq. (2.7), to produce $(U, V^{'})$.
6: **Return** $U, V^{'}$

In this algorithm, it is assumed that the data is sufficiently sampled. Therefore, the error between the cluster center locations produced by clustering the entire dataset and the location generated by clustering the sampled data should be small. The extension in step 5 of this algorithm used to calculate the full fuzzy data partition for any algorithm that approximates the cluster centers.

Nowadays, it has been observed that the accelerated and incremental clustering does not scale well for Big Data because the surging volume of Big Data makes the result of an accelerated and incremental clustering algorithm obsolete over time [40]. As mentioned in objectives, section 1.2, it is proposed to scale the accelerated and incremental clustering algorithms on Big Data processing frameworks. A survey of Big Data processing framework is presented next.

2.6 Big Data Frameworks and Related Algorithms

Today, a vast amount of digital data accumulated at an accelerating rate in many critical areas, which includes social networking, finance, health care, and bioinformatics. Due to the surging volume of Big Data from various sources,

there is a real pressing need for credible research into large-scale data analytics, to gain insights from the useful information in Big Data [40]. Clustering is a widely used data mining technique for Big Data analysis. However, the surging volume of Big Data has put tremendous pressure on clustering algorithms to scale beyond a single machine, due to both space and time bottlenecks. To scale the clustering algorithms for Big Data, there is a need for Big Data processing frameworks. In recent years, a large number of computing frameworks have been developed such as Hadoop, MapReduce (with its open-source implementations, such as Hadoop) [120], Apache Flink [121], Apache Mahout [122], Haloop [123], Apache Storm [124], Apache Samza [125], Hlive [126], Hbase [127], and Apache Spark [128,129]. The details of some of the Big Data processing frameworks are presented next.

2.6.1 Big Data frameworks

There are a wide variety of processing frameworks available for Big Data. The details are as follows:

- **Apache Hadoop**: It is a framework for distributed processing and distributed storage of VL datasets across multiple nodes using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. The core of Apache Hadoop consists of a storage part, known as HDFS, and a processing part called MapReduce. It splits files into large blocks and distributes them across nodes in a hadoop cluster. To process data in parallel, it transfers packaged code for nodes based on the data that needs to be handled.

- **MapReduce**: It is a framework for easily writing applications which process a vast amount of data in parallel on large clusters in a reliable and fault-tolerant manner. A MapReduce job usually splits the input dataset into independent chunks which are processed by the map tasks in a completely parallel manner. This framework sorts the outputs of the maps, which are then given as inputs to the reduce tasks. Typically, both the input and the output of the job are stored in a file-system.

- **Apache Flink**: It is an open source framework for distributed Big Data analytics, like Hadoop. The core of Apache Flink is a distributed streaming data flow engine, and it provides facilities for distributed computation over streams of data. It aims to bridge the gap between MapReduce-like systems and shared-nothing parallel database systems.

- **Apache Mahout**: It is an open source project that is primarily used in producing scalable machine learning algorithms and focused in the areas of collaborative filtering, clustering, and classification. It uses the Apache Hadoop library to scale efficiently in the cloud. The algorithms of Mahout are written on the top of Hadoop, so it works well in a distributed environment. It also offers the coder, a ready-to-use framework for doing data mining tasks on a large volume of data.

- **Apache Storm**: It is a distributed real-time computation system, designed for efficiently processing unbounded streams and can be used with any programming language. It can be utilized for real-time analytics, distributed machine learning, and numerous other cases, especially those of high data velocity. Storm can run on YARN and integrate into Hadoop ecosystems, providing existing implementations a solution for real-time stream processing. It is fast, scalable, fault-tolerant, reliable, and easy to operate Big Data processing framework.

- **Apache Haloop**: This framework is an extension of Hadoop which along with the processing of data provides an interesting way to perform iterative computation on the data. It provides inter-iteration locality where the major goal of haloop is to keep the data for map and reduce that uses same data in different iterations on the same machine. Here data is easily cached and is reused in various other applications.

- **Apache Samza**: It is another distributed stream processing framework. It uses Apache Kafka for messaging and Apache Hadoop YARN to provide fault tolerance, processor isolation, security, and cluster resource management.

- **Apache Spark**: It was originally developed at the University of California in 2009. It is an open source cluster computing framework built for so-

phisticated data analysis. Spark is optimized for iterative algorithms and large-scale interactive computation. Apache Spark works well for iterative algorithms by supporting in-memory computations, while retaining the scalability and fault tolerance of MapReduce. Spark performs up to 100 times faster than Hadoop MapReduce and significantly faster than other frameworks [128, 129].

As mentioned in the objectives, section 1.2, it is proposed to design the scalable clustering algorithms which aim to utilize the Apache Spark framework for the processing of Big Data. Therefore, a detailed description of Apache Spark framework is presented next.

2.6.2 Working of Apache Spark

Spark cluster consists of many machines, each of which is referred to as a node. There are two main components of Spark: Master and Workers. There is only one master node, and it assigns jobs to the slave nodes or worker nodes. However, there can be any number of slave nodes. Data can be stored in HDFS [130] or a local machine. A job reads data from the local machine/HDFS, performs a computation on it and writes some output. Spark can run over a variety of cluster managers. A simple cluster manager included in Spark itself is known as Standalone Scheduler, on Apache Mesos [131], or on Hadoop YARN [132]. For distributed data storage, the spark can interface with a wide variety including Hive [126], HBase [127], Tachyon [133], and HDFS [134]. Figure 2.2 describes how the master node divides the job among the worker nodes. A Driver process executing on the master node is responsible for running this job on the Spark Engine. Each job is partitioned into some stages. Each stage can either be a

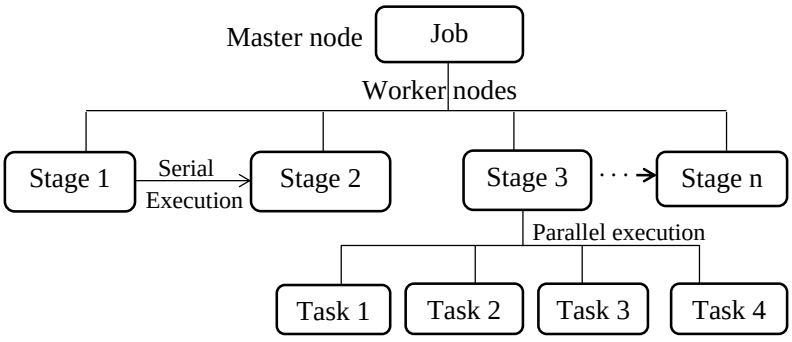


Figure 2.2: Internal workflow of Apache Spark.

map or a reduce stage represented by a map function and reduceByKey function which are discussed in detail in section 6.2.1, chapter 6. One stage may be dependent on the results of a previously occurring stage. All stages are executed serially. The dataset is logically divided into several chunks, and these chunks are assigned to different worker nodes. A task performs the set of operations, defined by a stage, on a chunk. These tasks are performed by executor processes on a worker node. Only one task is performed on one chunk by each executor.

Features of Apache Spark

Spark performs better than MapReduce, the features of Apache Spark is presented as follows:

1. **Speed** - Spark helps to run an application on a Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on a disk. Spark makes it possible by reducing the number of read/write operations to disk. It stores the intermediate processing data in-memory. It means that it does in-memory computations, i.e., it copies the data from the distributed physical storage to the much faster logical RAM. The in-memory computations feature help Spark to eliminate the disk I/O time, making it 100 times faster than Hadoop MapReduce [128, 129]. It uses the concept of an RDD, which allows it to transparently store data in memory and persist it to disk only if it is needed. The RDD helps in reducing read and write to disk which is one of the main time-consuming factors. The details of RDD are presented subsequently. Figure 2.3 describes the operations performed on Spark and shows that how intermediate data retained in RAM and makes its execution 100 times faster than Hadoop MapReduce.

2. **Supports multiple languages** - Spark provides built-in APIs in Java, Scala, and Python. It also has data frame APIs for manipulating semi-structured data. Its ability to support multiple languages makes it dynamic.

3. **Support for sophisticated analytics** - Spark not only supports 'map' and 'reduce' operations. It also supports SQL queries, streaming data, and complex analytics such as machine learning and graph algorithms.

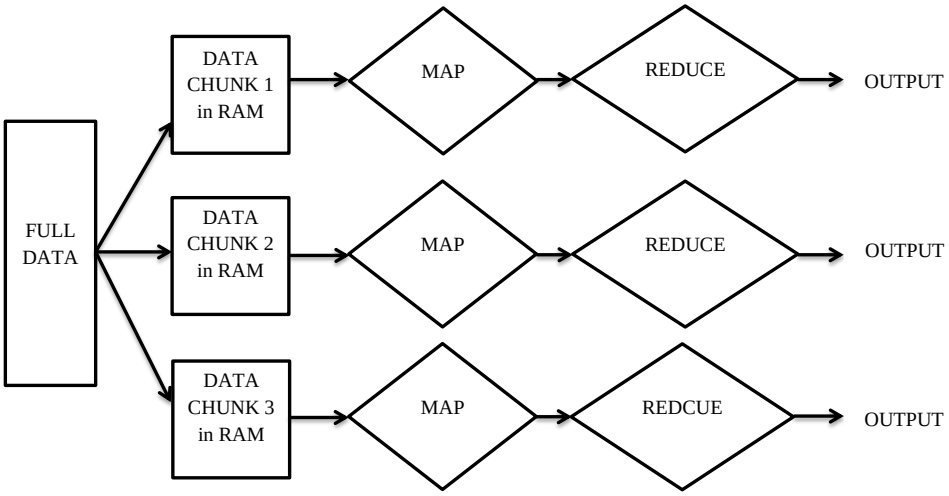


Figure 2.3: In-memory computation of Spark.

4. **Real time stream processing** - Spark can handle real-time stream processing along with the integration of other frameworks which concludes that sparks streaming ability is easy, fault tolerance and integrated.

5. **Ability to integrate with Hadoop and existing Hadoop Data** - Spark can run independently. In addition to this, it is compatible with both versions of Hadoop ecosystem, i.e., it can run on Hadoop YARN cluster manager [132] and SIMR. It can also read any existing Hadoop data that's makes it suitable for migration to pure Hadoop-MapReduce applications.

6. **Supports lazy evaluation** - Another outstanding feature of Spark is that it follows a lazy approach, i.e., it loads and pushes an RDD into the RAM only when an action needs to be performed, not when it encounters an RDD. This means that it waits for instructions before providing a final result which saves a significant time.

Resilient Distributed Datasets

RDD [128] is the representation of data in object format on which computations can be performed in parallel. Spark provides special operations on RDDs containing key-value pairs. These RDDs are called pair RDDs. It is a useful building block in many programs, as they expose operations that allow you to act on each key in parallel or regroup data across the network

with the same key. Two types of operations that can be performed on an RDD:

(a) Transformations: Transformations are operations on an RDD which result into another RDD.

- Map: It is a transformation that passes each data point through a function and returns a new RDD representing the results such as my_data.map(lambda x: 2x) gives an RDD where each element is twice the value of each element in my_data. Pair RDDs are allowed to use all the transformation available to standard RDDs. Pair RDDs can be created by running a map() function that return key-value pairs. For example, pairs=line.map(lambda x: (x.split(" ")[0],x)).
- Aggregations: When a dataset is described in terms of key-value pairs, it is common to aggregate statistics across all the elements with the same key. Spark has a set of operations that combines values with the same key by using a reduceByKey function. This function merges the values for each key using an associative reduce function. It works only for RDDs which contains key and value pairs kind of elements (i.e. RDDs having tuple or Map as a data element). In this, one associative function is passed as a parameter, which will be applied to the source RDD and will create a new RDD with resulting values (i.e. key-value pairs). This function produces the same result when repetitively applied on the same set of RDD data with multiple partitions irrespective of elements order.

(b) Actions: These operations are performed only when the program needs to answer a question. For example, my_data.count() counts the number of elements that are present in my_data.

2.6.3 Algorithms for Processing Big Data

Recently, a wide variety of algorithms [50, 135–138] has been proposed by researchers for processing Big Data using various frameworks. This is because the enlarging volume of information emerging by the progress of technology makes

the clustering of Big Data a challenging task. Kwok et al. [135], proposed a parallel version of the FCM algorithm for clustering. The designed algorithm runs on parallel computers of the SPMD model with the MPI. Beringer et al. [136], developed a scalable online version of the Fuzzy C-Means algorithm. In this method, the aim is to consider the problem of clustering of continuously evolving time series data in the form of parallel streams of real value data. In particular, this approach seems to be useful for the applications, in which the clustering structure is subjected to continuous changes. Zhao et al. [50], proposed a parallel K-means clustering algorithm based on MapReduce, which is a simple yet powerful parallel programming technique. Zhang et al. [137], proposed i^2MapReduce, a novel MapReduce-based framework for incremental Big Data processing. It combines fine-grain incremental engine, a general-purpose iterative model, and a set of useful techniques for incremental iterative computation. It performs key-value pair level incremental processing and provides support for not only one-step calculation but also supports more sophisticated iterative computation. Furthermore, it incorporates a set of novel techniques to reduce I/O overhead for accessing preserved fine-grain computation states.

In recent scenario, Big Data is evolving in various domains, so it is important to investigate the applicability and performance of scalable Big Data algorithms on a real-life problem. As discussed in the objectives, section 1.2, it is proposed to study the applicability of scalable clustering algorithms on a real-life Big Data classification problem, i.e., the protein database of the complex plant genome. For working on this Big Data problem, there is a tremendous need to propose an efficient feature extraction approach for protein sequence classification. Therefore, a survey related to feature extraction approaches is presented next.

2.7 Real-life Plant Genome Data

Bioinformatics has been an active area of research for the last three decades, and it is continuously gaining careful attention from computer scientists and biologists research community. The objective of bioinformatics is to store and manage the biological data and to develop sophisticated computational tools that are helpful in analysis and modeling [139]. Genome sequencing projects are

currently producing an enormous amount of new sequences and cause the rapid increase of protein sequence databases. Protein sequences contain characters from the 20-letter amino acid alphabets $\sum$={A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. Protein sequences are of any length, and the amino acids present in a sequence can be combined in any order. There is a substantial need of machine learning approaches for the analysis and modeling of an enormous amount of protein sequence data.

An important issue in applying any algorithm to protein sequence classification is how to represent the protein sequences in terms of feature vectors. The high dimensionality of protein data creates several crucial problems for the researchers during the implementation of machine learning algorithms [140]. Evidently, a good input representation (extraction of features) is necessary for the proper classification of protein sequences. Many feature selection techniques [141–149] have been introduced in the past, but still, there is a need of a method that can select statistically significant features for each protein sequence. The extraction of significantly useful features would increase the classification accuracy by removing the redundant or unnecessary features and also decrease the run-time of classification algorithms. The automated classification mechanism saves the long time required for the experiments and the expense of expensive biological tests performed in the laboratories.

Wang et al. [142] proposed a new technique for feature extraction which tries to capture both the global similarity and local similarity of protein sequences. The global similarity refers to the overall similarity among multiple sequences, whereas the local similarity refers to frequently occurring substrings in the sequences. It considered 2-gram method as discussed in [141] to compute the global similarity of protein sequences and adopted a method called 6-letter exchange groups to represent a sequence [150]. Then it uses sequence mining tool to compute the local similarity.

Leslie et al. [143] proposed a spectrum kernel to measure the sequence similarity between protein sequences. The technique considered subsequences of k length amino acids as a feature vector. The obtained feature vector was then passed to a support vector machine for classification of protein sequences into their relevant classes. Another feature extraction approach is proposed by Bandyopadhyay [144] that uses 1-gram technique for feature encoding. The fea-

ture size comprises of 20 amino acids. The extracted features are such that they take into consideration the probabilities of occurrences of the amino acids in the different positions of the sequences.

Mansoori et al. [145], proposed a new technique for feature extraction. To extract the relevant features from protein sequences, the features are counted as the occurrences of six exchange groups [150] in each sequence. Another feature extraction approach for protein sequence is proposed by Mansoori et al. [146]. This method uses 2-grams and a 2-gram exchange group from the training and test data. In this approach, distance-based feature ranking method was utilized for the selection of the best and most appropriate features.

Caragea et al. [147] investigated feature hashing technique to map high-dimensional features to low-dimensional using hash keys. The high dimensional features were obtained using the traditional k-gram representation. The feature vector obtained by applying hash function stores frequency counts of each K-gram hashed together in the same hash key. With this technique, multiple features can be mapped to the same key. The size of the feature vector has been reduced from 2^{22} to 2^{10} without a major decrease in the classification accuracy.

Yu et al. [148] proposed a k-string dictionary technique to represent a protein sequence. The value of "k" can be 1, 2,. . ., k amino acids. Repeated k-strings were considered only once. The frequency or probability of each k-string vector was observed during the experiments. Then SVD was applied for the factorization of frequency/probability matrix, representing each protein sequence correctly. This technique reduces the size of the feature vector significantly.

Iqbal et al. [149] proposed a statistical metric-based feature subset selection technique for the selection of discriminant and invariant features from protein sequences. In this method, during the sequence encoding process, each protein sequence is represented by n-gram descriptors frequency. The statistical metric is used in this approach to discard the irrelevant or redundant features and represent the protein sequences with a minimum number of statistically discriminant features.

2.8 Performance Measures

This section presents the different performance measures to evaluate the proposed clustering algorithms. To assess the performance of fuzzy clustering, cluster validity index is one of the important factor for all sorts of fuzzy clustering algorithms as it determines the number of clusters required for grouping of different datasets by fuzzy clustering. The details about this are already presented in section 2.2. Furthermore, to evaluate the performance of fuzzy clustering, a novel cluster validity index is proposed and discussed in detail in Chapter 3 that determines the optimal number of clusters for different datasets. However, there are other factors, i.e., fuzziness parameter (m) and a set of initial cluster centers which affects the performance of fuzzy clustering. Many hybrid approaches are proposed by the researchers [24, 25, 27, 42, 43, 74, 82, 103] to determine these factors, which are already discussed in detail in section 2.3 and section 2.4. Moreover, two novel hybrid approaches are proposed and reviewed in depth in Chapter 4 to determine these factors for the investigations of improvement in the performance of fuzzy clustering. In addition to this, for evaluating the performance of scalable clustering algorithms, various measures are used such as NMI [46], F-measure [45], ARI [47,48], speedup, sizeup, scaleup [49,50], and Objective function. Also, the trade-off between performance gain versus accuracy is also evaluated for in-depth analysis of the performance of scalable clustering algorithm. The detail of these measures is present in the subsequent section. After this, we discuss the measures used to evaluate the performance of incremental clustering algorithm and the proposed feature extraction approach. To do this, we evaluate the average classification accuracy, specificity, sensitivity, and confusion matrix [151]. The details of various measures are presented next.

2.8.1 Performance Measures for Scalable Algorithms

Here, we discuss the measures used to evaluate the performance of the proposed scalable fuzzy clustering algorithms on Big Data which are as follows:

- **NMI** [46]

  It is used to compute the clustering results, which measure the agreement of the clustering results produced by an algorithm and the ground truth.

If we refer to the class as the ground truth and to cluster as the results of a clustering algorithm, the NMI is calculated as follows:

$$NMI = \frac{\sum_{j=1}^{c}\sum_{l=1}^{w} n_j^l log\big(\frac{n.n_j^l}{n_j.n_l}\big)}{\sqrt{\big(\sum_{j=1}^{c} n_j log\big(\frac{n_j}{n}\big)\big)\big(\sum_{l=1}^{w} n_l log\big(\frac{n_l}{n}\big)\big)}} \tag{2.23}$$

where w is the number of classes, n_j are the number of data points belongs to j^{th} cluster and n_l are the number of data points belongs to l^{th} class, respectively, and n_j^l is the number of common data points belongs to both the l^{th} class and j^{th} cluster.

- **F-measure** [45]

  It helps in determining the accuracy of a clustering solution. The F-measure $F(j,l))$ for a j^{th} cluster with respect to a particular l^{th} class indicates how good j^{th} cluster describes l^{th} class by calculating the harmonic mean of precision and recall as follows:

$$p_j^l = \frac{n_j^l}{n_l}, r_j^l = \frac{n_j^l}{n_j} \tag{2.24}$$

$$F(j,l) = \frac{2 * p_j^l * r_j^l}{p_j^l + r_j^l} \tag{2.25}$$

  where precision p_j^l is the fraction of n_j^l and n_l, recall r_j^l is the fraction of n_j^l and n_j. The overall F-measure is defined as:

$$F - measure = \sum_j \frac{n_j^l}{n} max_l F(j,l) \tag{2.26}$$

  A higher value of F-measure indicates better clustering results. F-measure equal to 1 indicates that the clustering result is same as the ground truth.

- **ARI** [47, 48]

  It is the corrected-for-chance version of the Rand index, which evaluates the similarity between two partitions. The ARI measure assumes that the clustering is discrete, i.e., hard [152]. To compute the ARI, in the case of fuzzy clustering we first harden the fuzzy partitions by setting the highest membership value of each data point to the cluster equal to 1, and all else to 0 [36]. We use ARI to compare the clustering solutions with ground

truth labels (when available), as well as it examines the partition of an accelerated algorithm to that of the reference algorithm. The formulation of ARI is defined as follows:

$$ARI = \frac{\sum_{j=1}^{c}\sum_{l=1}^{w}\binom{n_j^l}{2} - \left[\sum_{j=1}^{c}\binom{n_j}{2}\sum_{l=1}^{w}\binom{n_l}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{j=1}^{c}\binom{n_j}{2} + \sum_{l=1}^{w}\binom{n_l}{2}\right] - \left[\sum_{j=1}^{c}\binom{n_j}{2}\sum_{l=1}^{w}\binom{n_l}{2}\right]/\binom{n}{2}} \quad (2.27)$$

- **Speedup** [49, 50]

  It is the measure to compute the performance of the algorithm in terms of run-time by keeping the dataset constant and varying the number of worker nodes in the spark cluster.

- **Sizeup** [49, 50]

  It computes the performance of the algorithm in terms of run-time by keeping the number of worker nodes in the spark cluster constant and varying the size of the dataset.

- **Scaleup** [49, 50]

  It computes the performance of the algorithm in terms of run-time by varying both the number of worker nodes and the size of the dataset. It is the ability of an m-times larger system to perform an m-times larger job in the same run-time as the original system.

- **Objective Function Value**

  The performance of scalable clustering algorithms is evaluated by its ability to minimize the objective function stated in Eq. (2.6).

Apart from the performance measure discussed above, there is a need to evaluate the trade-off between the performance gain (in terms of run-time) versus accuracy (in terms of NMI) for accurate assessment of scalable clustering algorithms. The trade-off is evaluated in terms of percentage change in run-time and the loss of accuracy (in terms of NMI) which shows that with the decrease of run-time how much the accuracy gets affected. The detailed description of the same is presented in section 6.4.2, and its experimental evaluation is presented in terms of Figure 6.8.

2.8.2 Performance Measures for Incremental Algorithm and Protein Feature Extraction Approach

The measures used to evaluate the performance of the proposed incremental clustering algorithm and the proposed protein feature extraction approach are presented as follows:

- **Accuracy**

  When the actual class labels are available for a test dataset, calculating the percentage accuracy of a clustering solution is an obvious metric, but somewhat misleading because clustering algorithms do not optimize efficiency. Each cluster label is associated with a class label and any data points whose cluster label does not match its associated class is considered inaccurate. Before the calculation, clusters must be aligned to the class labels that is done by using Bayes' theorem [153]. The average accuracy for multi-class classification [151] is defined as follows:

$$AverageAccuracy = \sum_{l=1}^{w} \frac{\frac{tp_l+tn_l}{tp_l+fn_l+fp_l+tn_l}}{w} \times 100 \tag{2.28}$$

  where, tp_l, tn_l, fn_l and fp_l represents the true positive, true negative, false negative and false positive of l^{th} class, respectively.

  - True positive (tp_l): It denotes the correct classification of positive data of the l^{th} class. In the case of multi-class classification problem, the diagonal element of the matrix represents the true positive for each class.
  - True negative (tn_l): It denotes the correct classification of negative data of the l^{th} class. In the case of multi-class classification problem, true negative for a particular A^{th} class is evaluated as follows:

$$tn_A = tp_B + E_{BD} + E_{DB} + tp_D \tag{2.29}$$

  - False positive (fp_l): It denotes the incorrect classification of negative data of l^{th} class that is classified as positive. The parameter in case of multi-class classification problem is evaluated for a particular A^{th}

class is defined as follows:

$$fp_A = E_{BA} + E_{DA} \tag{2.30}$$

– False negative (fn_l): It denotes the incorrect classification of positive data of l^{th} class that is classified as negative. In case of multi-class classification problem, false negative for a particular A^{th} class is evaluated as follows:

$$fn_A = E_{AB} + E_{AD} \tag{2.31}$$

- **Sensitivity** [151]

  An average per class effectiveness of a classifier to identify the fraction of positive data that are classified as positive.

$$Sensitivity = \frac{\sum_{l=1}^{w} \frac{tp_l}{tp_l + fn_l}}{w} \times 100 \tag{2.32}$$

- **Specificity** [151]

  An average per class effectiveness of a classifier to identify the fraction of negative data that are classified as negative.

$$Specificity = \frac{\sum_{l=1}^{w} \frac{tn_l}{tn_l + fp_l}}{w} \times 100 \tag{2.33}$$

- **Confusion Matrix** [151]

  A confusion matrix contains information about actual and predicted classifications performed by a classifier. The performance of such classifiers is commonly evaluated using the data in the matrix. A confusion matrix in case of three class problem is given in Table 2.2. The terms are provided in Table 2.2 have already been detailed earlier.

Table 2.2: Confusion matrix.

		Predicted class		
		C_A	C_B	C_D
Actual class	C_A	tp_A	E_{AB}	E_{AD}
	C_B	E_{BA}	tp_B	E_{BD}
	C_D	E_{DA}	E_{DB}	tp_D

The proposed cluster validity index and hybrid approaches are designed to investigate the parameters for improving the performance of fuzzy clustering. All these methods are evaluated on various benchmark datasets collected from UCI Machine Learning Repository [154]. The experimental evaluation of these methods on benchmark datasets is presented in Chapter 3 and Chapter 4. In addition to this, the performance of the proposed incremental clustering algorithm is tested on VL datasets collected from UCI repository, and the experimental evaluation of the proposed algorithm on VL datasets is presented in Chapter 5. Furthermore, the performance of the proposed scalable clustering algorithms is tested on several Big Datasets that is created by taking data from other sources [154–157]. The experimental evaluation of scalable clustering algorithms on several Big datasets is presented in chapter 6. The validation methods used for evaluating the results are shown in Appendix B. The details of various benchmark datasets and VL datasets are presented in Appendix A, whereas the details of Big datasets are presented next.

2.9 Working with Datasets

This section presents the details of datasets used in the experimental study of all the proposed approaches. In all, total fifteen datasets are utilized in the experiments out of which twelve datasets are taken from UCI Machine Learning repository [154] and remaining three Big Datasets are created by taking data from different sources [154–157]. The benchmark datasets namely IRIS, WINE, VEHICLE, GLASS, SEED, BUPA, DIABETES, PAGE BLOCKS, PENDIGITS, SUSY, HTRU2, and EEG Eye State are taken from UCI repository. The detailed description of these benchmark datasets are presented in Appendix A. Out of these twelve benchmark datasets, first six datasets are used to investigate the performance of proposed cluster validity index presented in chapter 3. Furthermore, the first four benchmark datasets along with DIABETES dataset are used to evaluate the performance of proposed hybrid approaches discussed in chapter 4. The performance of the proposed incremental clustering algorithm is evaluated on four VL datasets, i.e., PAGE BLOCKS, PENDIGITS, HTRU2, and EEG Eye State and the discussion for the same is presented in chapter 5. However, for the sake of testing of scalable clustering algorithms on Big Datasets,

we have simulated three datasets by taking them from various sources [154–157], to enlarge their volume in the range of 2.4 GB to 34 GB. The performance of scalable clustering algorithms is evaluated on four Big Datasets, i.e., the SUSY dataset along with three Big simulated datasets and the results for the same is presented in chapter 6. Table 2.3 lists the number of samples, features, classes, and size of these Big Datasets. The detailed description of these Big Datasets is presented next.

SUSY

The dataset is taken from UCI Machine Learning repository [154]. It has been produced using Monte Carlo simulations. The size of this dataset is 2.4 GB and contains 5,000,000 instances which are divided into 2 classes and has 18 features. The first 8 features are kinematic properties measured by the particle detectors in the accelerator. The last ten features are functions of the first 8 features; these are high-level features derived by physicists to help discriminate between the two classes.

MONARCH-SKIN

The dataset is constructed by randomly sampling B, G, R values from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database. Total learning sample size is 2,45,057 having 3 features and consists of 2 classes; out of which 50,859 is the skin samples and 1,94,198 is non-skin samples.

To create a massive dataset, we did the Cartesian product of SKIN SEGMENTATION data obtained from UCI [154] with monarch dataset [156] having only one feature. A subset of the Cartesian product of 1,470,342,000 instances and size 25.04 GB is used as the dataset in the experimental study. It is appropriately named as Monarch-Skin dataset.

Table 2.3: Information of datasets used for Big Data experimentation.

Datasets	Samples	Features	Classes	Size
SUSY	5,000,000	18	2	2.4 GB
MINST8m	8,100,000	784	10	19 GB
Monarch-Skin	1,470,342,000	3	2	25.4 GB
Replicated-USPS	14,582,000	256	10	34 GB

REPLICATED-USPS

It is a numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The originally scanned digits are binary and of different sizes and orientations; the images here have been deslanted, and size normalized, resulting in 16 x 16 grayscale images. The original dataset consists of 7291 training samples and 2007 test samples, each having 256 features and consists of 10 classes [155].

To create an enormous amount of data, we replicate the original USPS dataset 2000 times to make it larger and named it as Replicated-USPS dataset. The resulting size of the dataset is 34 GB and consists of 14,582,000 instances. This is done on the basis of work done by researchers recently [137, 138].

MINST8m

It is a handwritten digit recognition dataset of size 19 GB and contains 8,100,000 greyscale images (28x28 pixels of the digits 0 to 9) [157]. Each pixel has an integer value between 0 to 255. The pixel values are scaled between [0,1] by dividing it by 255, and each image is represented as a 784-dimensional feature vector. It consists of total 10 number of classes.

Chapter 3

Cluster Validity Index

3.1 Introduction

One of the most widely used fuzzy clustering algorithms is FCM proposed by Bezdek [22]. In this algorithm, for the computations of fuzzy partitions, it is required that the user should pre-define the number of clusters (c). Since the optimality of fuzzy c-partitions are dependent on the choice of c, so it is important to validate the fuzzy c-partitions, to determine the optimal number of clusters and the corresponding best fuzzy partition [74]. The validation of the fuzzy c-partitions is done by a cluster validity index, which measures the quality of c discovered clusters [88, 158] in each fuzzy partition. Thus, the largest or smallest value of a cluster validity index among all the fuzzy partitions gives the optimal number of clusters and the corresponding best fuzzy partition. Cluster properties such as compactness and separation are often considered as two primary criteria in various cluster validity indexes for evaluating clustering results and for the selection of an optimal clustering scheme [159]. The compactness measure evaluates the density of data points that belongs to the same cluster, whereas the separation measure evaluates the isolation among clusters. In addition to compactness and separation measures, an overlap measure is used in many cluster validity indexes [95, 98], to evaluate the degree of overlap of a specified number of fuzzy clusters.

The classical validity indexes validate the fuzzy partitions by using either of the two measures out of three. They are limited in their ability as they do not jointly exploit all the three measures, which leads to an incorrect validation

result [95]. In this chapter, we proposed a novel cluster validity index termed as VI_{DSO}, which jointly exploits all the three measures, i.e., compactness as dispersion, separation, and overlap, to find the optimal number of clusters and the corresponding best fuzzy partition. Insights of the proposed cluster validity index are presented next.

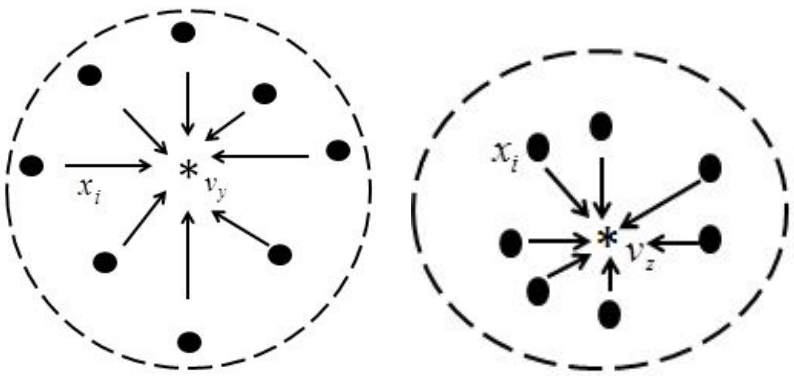
3.2 Proposed Novel Cluster Validity Index

In this section, a novel cluster validity index is designed by exploiting three proposed measures, i.e., intra-cluster compactness, inter-cluster separation, and inter-cluster overlap. The detailed description of these measures is presented below.

3.2.1 Intra-cluster Compactness

The proposed intra-cluster compactness measure ($Disp(c, U)$)), is computed by the relative variability concept known as the coefficient of variation. It calculates the relative dispersion of data points in all the dimensions and considers the dimension in which data points have maximum dispersion and aims to minimize the maximum dispersion. Minimization of the maximum dispersion indicates that the overall dispersion of data points in other dimensions within the cluster is reduced. Figure 3.1, shows the two fuzzy clusters with different degree of compactness. The proposed definitions for the intra-cluster compactness measure is presented as follows:

Firstly, we evaluate the dispersion of all the data points in each dimension.



(a) $F_y \in (U_y, V_y)$ (b) $F_z \in (U_z, V_z)$

Figure 3.1: Two fuzzy clusters F_y and F_z with different degree of compactness.

The small value of this term indicates that the data points are tightly coupled with each other in all the dimensions. The definition for the dispersion is given as follows:

Definition 1: Standard deviation of data points present in set X in f^{th} dimension is denoted by $\sigma(X_f)$ and is defined as:

$$\sigma(X_f) = \sqrt{(\sum_{i=1}^{n}(x_{if})^2 - (\mu(X_f) \times n)^2/n)/n} \tag{3.1}$$

where, n is the number of data points, $X_f = \{x_{1f}, ..., x_{nf}\}$; $\forall x_i \in \mathcal{R}^d$, $f = 1, 2, ..., d$ such that $\sigma(X) = \{\sigma(X_1), ..., \sigma(X_d)\}$; $X \in \mathcal{R}^d$ denote the data points present in set X consist of d dimensions; $\mu(X_f) = \sum_{i=1}^{n} x_{if}/n$ indicates the mean of n data points in f^{th} dimensions; $\forall x_i \in X$.

Next, we compute the relative dispersion of all the n data points in f^{th} dimension. The definition for the relative dispersion is presented as follows:

Definition 2: Coefficient of Variation of all the data points present in set X in f^{th} dimension is denoted by $Coff\_var(X_f)$ and is defined as:

$$Coff\_var(X_f) = \frac{\sigma(X_f)}{\mu(X_f)} \tag{3.2}$$

where, $Coff\_var(X_f) \in Coff\_var(X)$ such that $Coff\_var(X) = \{Coff\_var(X_1)$ $, ..., Coff\_var(X_d)\}$.

Next, we compute the dispersion of each cluster in d dimensions. The small value of this term indicates that the dispersion within the c number of clusters in each dimension is minimized. The definition for the same is given as follows:

Definition 3: Standard deviation of n data points on j^{th} cluster in f^{th} dimension is denoted as $\sigma_{v_{fj}}$ and given by:

$$\sigma_{v_{fj}} = \sqrt{\frac{1}{n}[\sum_{i=1}^{n}(x_{if} - v_{fj})^2]} \tag{3.3}$$

where $\sigma_{v_j} = \{\sigma_{v_{1j}}, ..., \sigma_{v_{dj}}\}$, $v_j \in \mathcal{R}^d$, $j = 1, 2, ..., c$, $f = 1, 2, .., d$, $X = \{x_1, .., x_i, .., x_n\}$, $\forall x_i \in \mathcal{R}^d$, v_{fj} denote the j^{th} cluster in f^{th} dimension, and d represents the number of dimensions.

Next, we evaluate the relative dispersion of each cluster in d dimensions. The

small value of this term indicates the particular dimension in which each cluster has lesser dispersion as compared to other dimensions. The definition for the same is presented as follows:

Definition 4: Coefficient of Variation of j^{th} clusters in f^{th} dimension is denoted as $Coff\_var_{v_{fj}}$ and is defined as:

$$Coff\_var_{v_{fj}} = \frac{\sigma_{v_{fj}}}{v_{fj}} \tag{3.4}$$

where, $Coff\_var_{v_{fj}} \in Coff\_var_{v_j}$ such that $Coff\_var_{v_j}$ =$\{Coff\_var_{v_{1j}}, ..., Coff\_var_{v_{dj}}\}$, $f = 1, 2, .., d$.

Definition 5: The overall dispersion within c number of clusters is defined as:

$$Disp(c, U) = \frac{\max\limits_{1\leq j\leq c} \max\limits_{1\leq f\leq d}\{Coff\_var_{v_{fj}}\}}{\max\limits_{1\leq f\leq d}\{Coff\_var(X_f)\}} \tag{3.5}$$

The numerator in the proposed formulation computes the dispersion of each cluster in d dimensions, respectively, and finally, it considers the cluster which has the maximum variation in a particular dimension. The denominator considers the dimension in which all the data points have a maximum dispersion in comparison to other dimensions. Therefore, evaluation of mathematical expression $Disp(c, U)$ considers the dimension in which the data points within the clusters have maximum dispersion relatively with other dimensions. Thus, it reflects the overall dispersion corresponding to all the data points within c clusters. Hence, the small value of $Disp(c, U)$ indicates the higher compactness within the cluster.

3.2.2 Inter-cluster Separation based on Fuzzy Set

The separation measure plays a significant role in evaluating the quality of fuzzy partitions produced by fuzzy clustering. Figure 3.2, shows the separation between the fuzzy clusters. The proposed inter-cluster separation ($Sep(c, U)$) quantifies the distance between the clusters using fuzzy set theory. To evaluate this, we utilize the similarity measure suggested by Lee [99]. The similarity between the two fuzzy cluster F_z and F_y at data point x_i is defined as follows:

$$S(F_z, F_y) = \max_{1\leq i\leq n} \min(u_{F_z}(x_i), u_{F_y}(x_i)); \forall x_i \in X \tag{3.6}$$

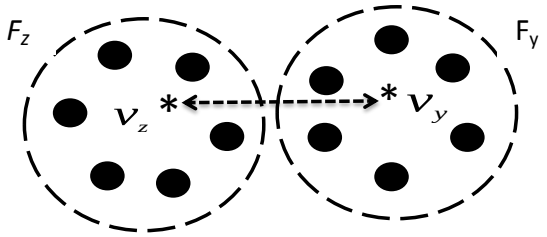


Figure 3.2: Two fuzzy clusters F_z and F_y shows the separation between cluster centers.

where $u_{F_z}(x_i)$ and $u_{F_y}(x_i)$ are the membership degrees of i^{th} data point corresponding to fuzzy clusters F_z and F_y, respectively. On the basis of similarity measures suggested by Lee [99], the proposed definitions for the inter-cluster separation ($Sep(c, U)$) are presented as follows:

Definition 1: The separation between the two fuzzy clusters F_z and F_y are defined as follows:

$$Dist(F_z, F_y) = 1 - S(F_z, F_y) \tag{3.7}$$

Definition 2: The overall separation among c clusters is defined as:

$$Sep(c, U) = \min(Dist(F_z, F_y)) \tag{3.8}$$

The inter-cluster separation measure considers the clusters with maximum similarity, which conversely results in consideration of the clusters with minimum separation. Therefore, we maximize the distance between the fuzzy clusters which has the minimum separation to ensure the overall maximum separation between the c clusters. Thus, the large value of $Sep(c, U)$ indicates that the clusters which are separated by minimum distance are far apart from each other. Conversely, it shows the large separation between other pairs of fuzzy clusters. Thus, it results in generation of the well separated fuzzy partitions.

3.2.3 Inter-cluster Overlap Measure

Overlapping plays a significant role in fuzzy cluster analysis which is computed by an inter-cluster overlap measure. The proposed inter-cluster overlap measure evaluates the overlap of each data point x_i between two fuzzy clusters F_z and F_y, which is represented by $R(x_i, F_z, F_y)$. A degree of overlap is assigned to each data point x_i depending on the belongingness of that data point to the

clusters [95], and it is denoted by $\delta(x_i)$. Vague data points are assigned a higher degree of overlap than distinctly classified data points. Various formal definitions are proposed for the inter-cluster overlap, which is given below.

Definition 1: The overlap of each data point x_i between the two fuzzy clusters F_z and F_y are defined by $R(x_i, F_z, F_y)$ and computed as follows:

$$R(x_i, F_z, F_y) = \begin{cases} \delta(x_i), & if\Big(Dom_{\min}(x_i) > 0 \;\&\; Dom_{\max}(x_i) < 1\Big) \\ 0.0, & Otherwise \end{cases} \tag{3.9}$$

Where,

$$Dom_{\min}(x_i) = \min(u_{F_z}(x_i), u_{F_y}(x_i)) \tag{3.10}$$

$$Dom_{\max}(x_i) = \max(u_{F_z}(x_i), u_{F_y}(x_i)) \tag{3.11}$$

The maximum and minimum degree of membership of a data point x_i are represented by $Dom_{\max}(x_i)$ and $Dom_{\min}(x_i)$, respectively. The data point x_i is considered vague, if $Dom_{\max}(x_i) \leq 0.5$, then the degree of overlap $\delta(x_i)$ of data point x_i between two fuzzy clusters F_z and F_y will be higher and assigned as $\delta(x_i) = 1$. Conversely, if the data point x_i is not vague, and its maximum degree of membership to a particular cluster is much higher in comparison to the other clusters, i.e., $Dom_{\max}(x_i)$>0.5 & $Dom_{\max}(x_i)$< 1, then the degree of overlap ($\delta(x_i)$) of data point x_i is uniformly distributed in the range of [0.1, 0.9]. Otherwise, if the data point x_i is classified to a particular cluster, i.e., $Dom_{\max}(x_i) = 1$, then its degree of overlap $\delta(x_i)$=0.

Definition 2: The total overlap between two pairs of fuzzy clusters F_z and F_y is defined as follows:

$$O(F_z, F_y) = \sum_{i=1}^{n} R(x_i, F_z, F_y) \tag{3.12}$$

Definition 3: The total overlap between all pairs of fuzzy clusters is represented by $Overlap(c, U)$ and is computed by considering the pair of fuzzy clusters having maximum overlap. The formulation for this is defined as follows:

$$Overlap(c, U) = \max_{z \neq y}(O(F_z, F_y)) \tag{3.13}$$

The fuzzy partitions and the value of c on which $Overlap(c, U)$ attain its minimum value, indicates that we minimize the maximum overlap between a pair of fuzzy clusters. If the maximum overlap is reduced, then it shows that the other pairs of fuzzy clusters will have lesser overlap. Thus, it achieves the best fuzzy partition and results in well-classified data points within the cluster.

3.2.4 Formulation of Proposed Validity Index

Once all the three proposed measures, i.e., $Disp(c, U)$, $Sep(c, U)$, and $Overlap(c, U)$ for cluster analysis, is defined. The proposed cluster validity index VI_{DSO} is formed by jointly exploiting all the three measures. These three measures are of varying scales, so it is necessary to conciliate it through the normalization approach. Thus, all the three measures are computed for different c values, i.e., c=$[c_{\min}, ..., c_{\max}]$; $c_{\min}$=2, $c_{\max}$=$\sqrt{n}$ [4, 74–76]. These measures for different c values are defined as follows:

$$Disp(c, U) = [Disp(2, U), ..., Disp(c_{\max}, U)] \tag{3.14}$$

$$Sep(c, U) = [Sep(2, U), ..., Sep(c_{\max}, U)] \tag{3.15}$$

$$Overlap(c, U) = [Overalp(2, U), ..., Overlap(c_{\max}, U)] \tag{3.16}$$

The maximum value of each measure is computed as follows:

$$Disp_{\max} = \max_{c_{\min} \leq c \leq c_{\max}} [Disp(c, U)] \tag{3.17}$$

$$Sep_{\max} = \max_{c_{\min} \leq c \leq c_{\max}} [Sep(c, U)] \tag{3.18}$$

$$Overlap_{\max} = \max_{c_{\min} \leq c \leq c_{\max}} [Overlap(c, U)] \tag{3.19}$$

All the three measures are normalized corresponding to each value of c with respect to their maximum values $Disp_{\max}$, $Sep_{\max}$, and $Overlap_{\max}$. The normalized values of these measures are represented as follows:

$$Disp^N(c, U) = \frac{Disp(c, U)}{Disp_{\max}} \tag{3.20}$$

$$Sep^N(c, U) = \frac{Sep(c, U)}{Sep_{\max}} \tag{3.21}$$

$$Overlap^N(c,U) = \frac{Overlap(c,U)}{Overlap_{\max}} \tag{3.22}$$

where, $Disp^N(c,U)$ represents the normalized value of dispersion of data points within the cluster, $Sep^N(c,U)$ represents the normalized value of separation between the fuzzy clusters and $Overlap^N(c,U)$ represents the normalized value of overlap of data points between fuzzy clusters for a fuzzy partition with a particular U and c; U denote membership matrix or fuzzy partition matrix. The above-discussed measures are combined to formulate the proposed validity index VI_{DSO}, which is defined as follows:

$$VI_{DSO}(c,U) = \frac{Disp^N(c,U)}{Sep^N(c,U)} + \frac{Overlap^N(c,U)}{Sep^N(c,U)} \tag{3.23}$$

The motivation for designing Eq. (3.23) in this way is to give the equal importance (weightage) to the compactness and separation measures as well as to the overlap and separation measures. The proposed validity index is evaluated for different values of c which is varying from $[c_{\min}, ..., c_{\max}]$. The value of c and the corresponding fuzzy partition U on which $VI_{DSO}(c,U)$ attains its minimum value is considered as the optimal c and the optimal fuzzy partition. The minimum value of $VI_{DSO}(c,U)$ indicates that the data points present within the cluster are tightly coupled with each other. Furthermore, the obtained fuzzy clusters are overlapped with a lesser degree and well separated from each other. The working of VI_{DSO} index in terms of the flowchart is present in Figure 3.3.

The step-wise procedure of VI_{DSO} index involved in the validation of a number of clusters and their corresponding fuzzy partitions are presented in the form of ***Algorithm 3.1***, which is given next.

Algorithm 3.1 Algorithm for Evaluation of Proposed $VI_{DSO}(c,U)$ Index

1: **Input** Dataset $X = \{x_1, x_2, ..., x_n\}$ and initialize fuzzy clustering parameters as mentioned in Table 2.1.
2: **Output** Optimal number of clusters (c) and membership matrix (U).
3: **Begin**
4: **for** $c := c_{\min}$ **to** $c_{\max}$ **step** 1 **do**
5: **Call and execute *Algorithm 2.2*** for each value of c.
6: **Compute** and store the value of compactness, Separation, and overlap measure, i.e., $Disp(c,U)$, $Sep(c,U)$, and $Overlap(c,U)$ measure using Eqs. (3.5), (3.8), and (3.13) for each value of c.
7: **end for**

Algorithm 3.1 (Continued)

8: **Compute** the normalized value of all the measures, i.e., compactness measure $Disp^N(c,U)$, separation measure $Sep^N(c,U)$, and overlap measure $Overlap^N(c,U)$ using Eqs. (3.20), (3.21), and (3.22) for all the values of c.

9: **Compute** the proposed validity index $VI_{DSO}(c,U)$ using Eq. (3.23) for all values of c where, $c = [c_{\min}, ..., c_{\max}]$.

10: **Find** the optimal number of clusters (c) and the corresponding membership matrix (U) by storing the value of c that minimizes VI_{DSO}, which is computed as follows:

$$VI^{\min}_{DSO}(c,U) = \min_{c_{\min} \le c \le c_{\max}} [VI_{DSO}(c,U)] \tag{3.24}$$

11: **Return optimal** c **and** U**.**

12: **End**

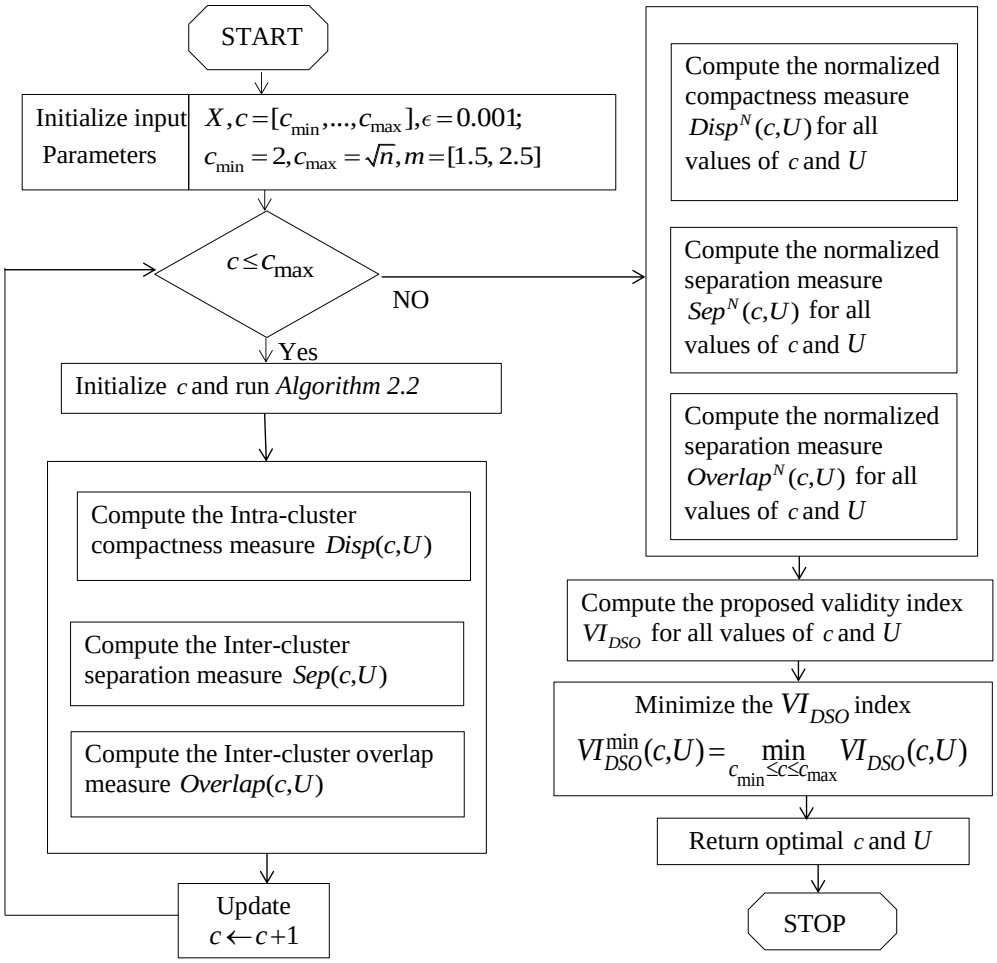


Figure 3.3: Flow chart of proposed cluster validity index.

3.3 Experimental Evaluation

In this section, we describe the datasets and the experimental settings used to perform our experimentation. Also, we present the experimental results to com-

pare our proposed cluster validity index with other well-known validity indexes namely Partition Coefficient (V_{PC}) and Partition Entropy (V_{PE}) [91], Rezaee et al. index (V_{CWB}) [93], and Kim and Lee index (v_{OS}) [95].

3.3.1 Datasets and Experimental Settings

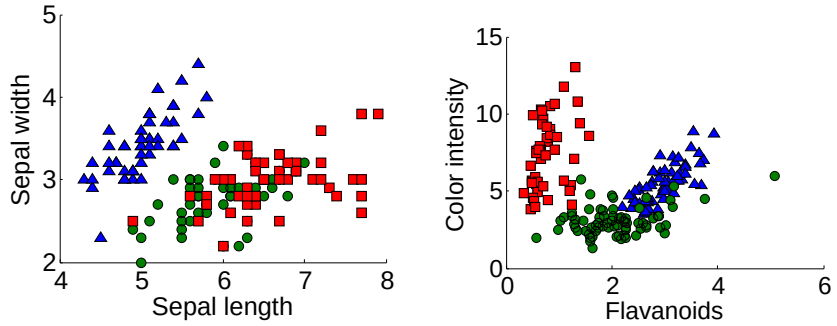
To demonstrate the effectiveness of our approach over other well-known cluster validity indexes, we have taken six datasets namely IRIS, WINE, VEHICLE, SEED, GLASS, and BUPA from UCI Machine Learning repository [154]. The detailed description of these datasets is given in Appendix A and scatter plot of these datasets is shown in Figure 3.4. In this figure, within the scatter plot of each dataset, a particular color represents a class, and the collection of data points shown by the same color belongs to a particular class. We implement all these validity indexes using MATLAB (Matlab R2014a) on an Intel(R) Core(TM) i3-2100 computer of 3.10GHZ with 2GB of RAM. The parameter values to implement all these validity indexes are given in Table 2.1.

3.3.2 Experimental Results and Discussion

In this section, we compare our proposed cluster validity index with other well-known validity indexes on six different datasets. In addition to this, we have also investigated the reliability of a VI_{DSO} index in comparison with other indexes with a change in fuzzifier (m).

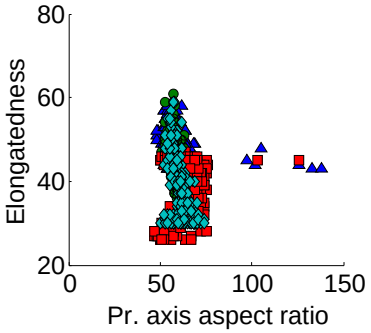
Comparison of proposed validity index with other indexes on IRIS dataset

In this section, we compare our proposed cluster validity index with other validity indexes on IRIS dataset. The scatter plot of this dataset is presented in Figure 3.4(a). The IRIS dataset consists of three classes representing three physical clusters. Therefore, in this figure, three different colors are used to represent three categories. Furthermore, all the data points which belong to the same class is represented by a particular color. Figure 3.4(a) shows that, according to their numerical representation out of three clusters, two clusters have the substantial overlap, while the third cluster is well separated from the other two. The proposed cluster validity index is evaluated on different values of

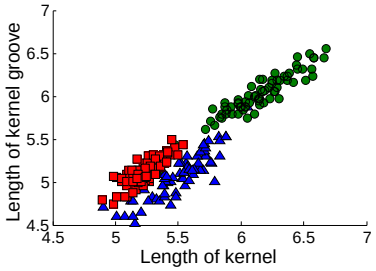


(a) IRIS Dataset

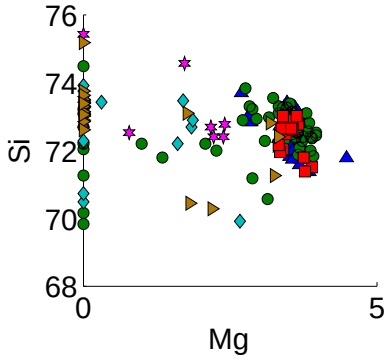
(b) WINE Dataset



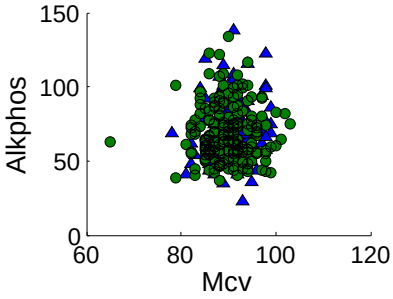
(c) VEHICLE Dataset



(d) SEED Dataset



(e) GLASS Dataset



(f) BUPA Dataset

Figure 3.4: Scatter plot of datasets in two dimensional space.

c ($c = [c_{\min}, ..., c_{\max}]$, $c_{\max} = \sqrt{n} \approx 12$) and found $c = 2$ as the optimal number of clusters. The same is being reported in Figure 3.5. The optimal number of clusters is found at $c = 2$ because the proposed index measures the overlap between the fuzzy clusters, and thus the data points with different class labels that are not distinguishable to each other by their features are grouped into a single cluster. Therefore, rather than considering three separate clusters, the overlapped data points are grouped into a single cluster [5]. Thus, we consider $c = 2$ as the optimal number of clusters according to the geometric structure of IRIS data as mentioned by Pal and Bezdek [74]. The proposed VI_{DSO} index is compared with other validity indexes, i.e., V_{PC} [91], V_{PE} [91], and V_{CWB} [93].

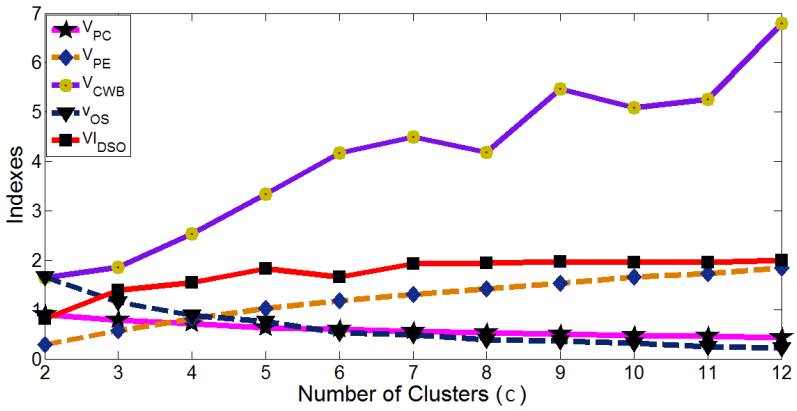


Figure 3.5: Comparison of cluster validity indexes on IRIS dataset at $m = 2.0$.

These indexes also find the optimal number of clusters at $c = 2$. On the other hand, v_{OS} index [95] attains the minimum value at $c = 12$. The v_{OS} index is monotonically decreases with c, which makes it hard to detect the optimal number of clusters. Thus, the optimal number of clusters for IRIS dataset is chosen to be 2.

Comparison of proposed validity index with other indexes on WINE dataset

In this section, we present the optimal number of clusters identified by the proposed validity index in comparison with other indexes on WINE dataset. As shown in Figure 3.4(b) according to the numerical representation two out of three clusters have the substantial overlap. The proposed validity index is evaluated for different values of c ($c = [c_{\min}, ..., c_{\max}]$, $c_{\max} = \sqrt{n} \approx 13$) it indicates the optimal number of clusters at $c = 2$, as shown in Figure 3.6. As discussed earlier, the proposed index finds the optimal clusters such that overlap between the fuzzy clusters is minimized. So rather than considering three separate clusters for overlapped data points (i.e., the data points with different classification labels that are not distinguishable to each other by their features), these data points are grouped into one cluster [5]. Thus, it achieves the optimal number of clusters at $c = 2$. The proposed VI_{DSO} index when compared with the existing indexes, i.e., V_{PC} [91], V_{PE} [91], and V_{CWB} [93], they also indicate the optimal number of clusters at $c = 2$. On the contrary, v_{OS} index [95] indicates that the minimum value reached at $c = 12$. As discussed earlier, v_{OS} index monotonically decreases with the increase of c, which makes it unreliable in detecting the optimal number of clusters for WINE dataset. Thus,

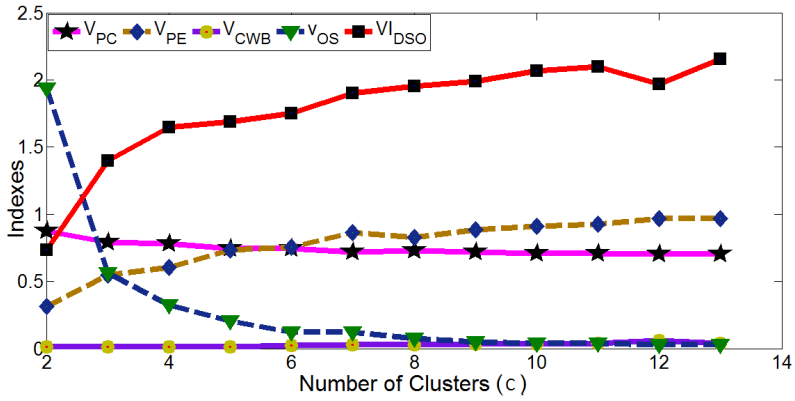


Figure 3.6: Comparison of cluster validity indexes on WINE dataset at $m = 2.0$.

an optimal number of clusters for WINE dataset is rated to be at $c = 2$.

Comparison of proposed validity index with other indexes on VEHICLE dataset

In this section, the proposed cluster validity index VI_{DSO} is compared with other cluster validity indexes in terms of an optimal number of clusters identified for VEHICLE dataset. Figure 3.4(c), shows the scatter plot of VEHICLE dataset. In this figure, the spread of data in two-dimensional space indicates the presence of three clusters. As shown in Figure 3.7, the performance of VI_{DSO} index when evaluated on different values of $c = [c_{\min}, ..., c_{\max}]$ ($c_{\max} = \sqrt{n} \approx 29$) it achieves the optimal value at $c = 3$. The VI_{DSO} index when compared with V_{PC} [91] and V_{PE} [91], both the indexes attains the optimal value at $c = 2$. Similarly, the V_{CWB} index [93] attain the minimum at $c = 5$ and v_{OS} index [95] attain the minimum at $c = 29$. In cluster analysis, compactness, separation, and overlap measures play a significant role, and the proposed VI_{DSO} index integrates all the

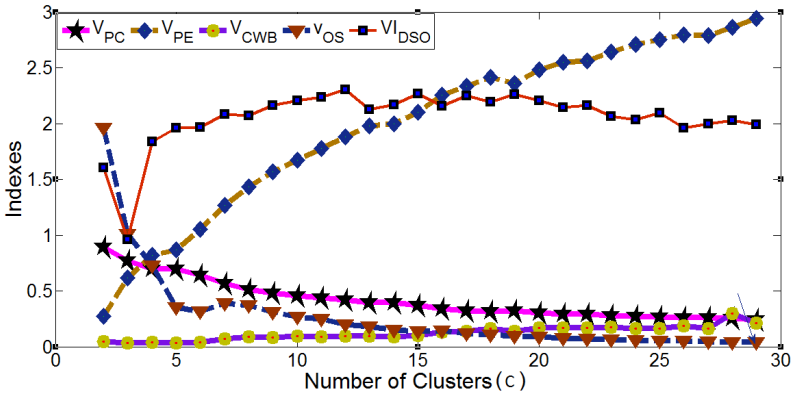


Figure 3.7: Comparison of cluster validity indexes on VEHICLE dataset at $m = 2.0$.

three measures. Therefore, the identified number of clusters $c = 3$ satisfies all the three properties, so $c = 3$ could be rated as the optimal number of clusters in comparison with other indexes for VEHICLE dataset.

Comparison of proposed validity index with other indexes on SEED dataset

In this section, we present the optimal number of clusters identified by VI_{DSO} index in comparison with other indexes on SEED dataset. The scatter plot of SEED dataset is presented in Figure 3.4(d). In Figure 3.8, we have evaluated the performance of all the validity indexes on a SEED dataset for different values of $c = [c_{\min}, ..., c_{\max}]$; $c_{\max} = \sqrt{n} \approx 14$. It is observed that the proposed VI_{DSO} index achieves the optimal number of clusters at $c = 3$. Similarly, V_{CWB} attains the optimal value at $c = 3$. On the contrary, the V_{PC} and V_{PE} indexes [91] reaches the minimum at $c = 2$, and v_{OS} index [95] indicates its minimum at $c = 14$. According to the reported results, the best partitioning of the data is achieved with 3 clusters so $c = 3$ could be considered as the optimal number of clusters for SEED dataset.

Comparison of proposed validity index with other indexes on GLASS dataset

In this section, the VI_{DSO} index is compared with the other cluster validity indexes in terms of the optimal number of clusters identified for GLASS dataset. The scatter plot of GLASS dataset is presented in Figure 3.4(e). The performance of VI_{DSO} index is evaluated by executing it for different values of $c = [c_{\min}, ..., c_{\max}]$; $c_{\max} = \sqrt{n} \approx 14$ and identified that it achieves the optimal

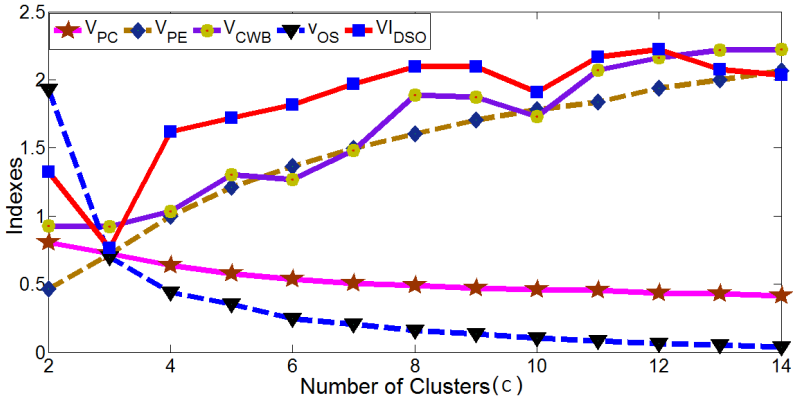


Figure 3.8: Comparison of cluster validity indexes on SEED dataset at $m = 2.0$.

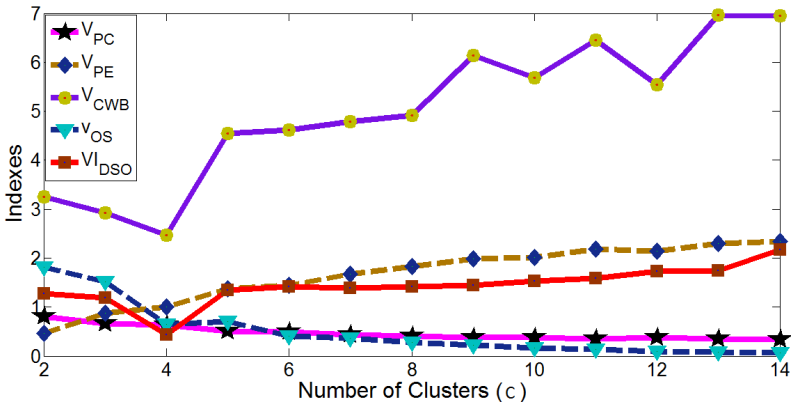


Figure 3.9: Comparison of cluster validity indexes on GLASS dataset at $m = 2.0$.

value at $c = 4$. The similar observation can be drawn from Figure 3.9, and it clearly shows the presence of 4 clusters with higher compactness, lesser overlap, and maximum separation. The performance of VI_{DSO} index when compared with V_{CWB} index [93], it shows that it achieves the optimal value at $c = 4$. On the other hand, the V_{PC} and V_{PE} indexes [91] attain the minimum value at $c = 2$ and v_{OS} index [95] indicating the minimum value at $c = 14$. The main problem with V_{PE} index is that it monotonically increases with c whereas the V_{PC} and v_{OS} indexes monotonically decreases with c, so unable to capture the underlying structure of dataset appropriately. Therefore, the optimal number of clusters cannot be rated based on these three validity indexes. Thus, the optimal number of clusters identified by VI_{DSO} and V_{CWB} indexes can be only measured at $c = 4$ for GLASS dataset.

Comparison of proposed validity index with other indexes on BUPA dataset

In this section, we discuss the optimal number of clusters identified by the proposed VI_{DSO} index in comparison with other cluster validity indexes for BUPA dataset. The scatter plot of BUPA dataset in two-dimensional space is shown in Figure 3.4(f). In Figure 3.10, the performance of VI_{DSO} index is evaluated for different values of $c = [c_{\min}, ..., c_{\max}]$; $c_{\max} = \sqrt{n} \approx 18$ and it is identified that the optimal value is reached at $c = 2$. The VI_{DSO} index when compared with the other indexes, i.e., V_{PC} and V_{PE}, they also attain the optimal value at $c = 2$. On the other hand, the other two indexes, i.e., V_{CWB} and v_{OS} achieves the minimum value of c at 5 and 18. Figure 3.4(f), clearly indicates the presence of two clusters with maximum compactness, lesser overlap, and higher

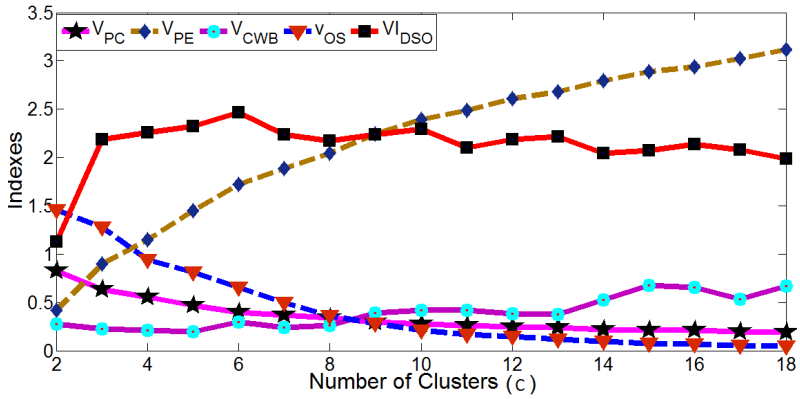


Figure 3.10: Comparison of cluster validity indexes on BUPA dataset at $m = 2.0$.

separation. Thus, the optimal number of clusters identified by VI_{DSO}, V_{PC}, and V_{PE} index is reached at $c = 2$.

Reliability analysis of proposed validity index in comparison with other indexes

As suggested by Pal and Bezdek [74], the reliability of validity index is judged when the optimal number of clusters identified by the validity index remains unaffected with the change in m values. Pal and Bezdek also suggested for $m \in [1.5, 2.5]$, the fuzzy clustering algorithm generates the best results. In this section, we have reported the results by investigating the reliability of VI_{DSO} index in comparison with other indexes for six different datasets. The results in terms of an optimal number of clusters are reported for each value of $m \in [1.5, 2.5]$ with a step size of 0.2 by varying c between $c_{\min}$ to $c_{\max}$. Tables 3.1(a), 3.1(b), and 3.1(f) present the results of IRIS, WINE, and BUPA datasets. The results emphatically show that for all the values of m, the VI_{DSO}, V_{PC}, and V_{PE} are the only indexes which correctly identify the presence of an optimal number of clusters at $c = 2$. On the other hand, the number of clusters reported by the V_{CWB} and v_{OS} indexes changes with a change in m. The results show that the proposed index in addition to other two indexes, i.e., V_{PC} and V_{PE} are considered reliable for IRIS, WINE, and BUPA datasets over other compared indexes. The similar observation is drawn in case of VEHICLE, SEED, and GLASS datasets from Tables 3.1(c), 3.1(d), and 3.1(e) which shows that VI_{DSO} is the only index that correctly identifies the optimal number of clusters for all three datasets on different values of m in comparison with other indexes. Furthermore, the optimal

Table 3.1: Identification of the optimal number of clusters (c) for different values of $m \in [1.5, 2.5]$ with a step size of 0.2.

(a) IRIS

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	**2**	**2**	2	12	**2**
1.7	**2**	**2**	2	12	**2**
1.9	**2**	**2**	2	12	**2**
2.1	**2**	**2**	2	12	**2**
2.3	**2**	**2**	3	12	**2**
2.5	**2**	**2**	3	12	**2**

(b) WINE

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	**2**	**2**	2	11	**2**
1.7	**2**	**2**	2	13	**2**
1.9	**2**	**2**	2	12	**2**
2.1	**2**	**2**	2	13	**2**
2.3	**2**	**2**	4	13	**2**
2.5	**2**	**2**	4	13	**2**

(c) VEHICLE

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	2	2	3	29	**3**
1.7	2	2	3	29	**3**
1.9	2	2	3	29	**3**
2.1	2	2	6	29	**3**
2.3	2	2	20	29	**3**
2.5	2	2	22	29	**3**

(d) SEED

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	2	2	2	14	**3**
1.7	2	2	2	14	**3**
1.9	2	2	2	14	**3**
2.1	2	2	3	14	**3**
2.3	2	2	3	14	**3**
2.5	2	2	3	14	**3**

(e) GLASS

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	2	2	4	13	**4**
1.7	2	2	3	14	**4**
1.9	2	2	4	14	**4**
2.1	2	2	4	14	**4**
2.3	2	2	4	14	**4**
2.5	2	2	4	14	**4**

(f) BUPA

m	V_{PC}	V_{PE}	V_{CWB}	v_{os}	VI_{DSO}
1.5	**2**	**2**	3	18	**2**
1.7	**2**	**2**	3	18	**2**
1.9	**2**	**2**	5	18	**2**
2.1	**2**	**2**	5	18	**2**
2.3	**2**	**2**	7	18	**2**
2.5	**2**	**2**	11	18	**2**

number of clusters identified by proposed index does not change with a change in m values. Therefore, VI_{DSO} index is considered as more effective and reliable over other indexes.

3.4 Summary

In this chapter, we have presented the design of novel cluster validity index, which integrates the three proposed measures namely intra-cluster compactness, inter-cluster separation based on the fuzzy set, and inter-cluster overlap. The purpose of this work is to find an optimal number of clusters and the corresponding fuzzy partition for the validation of fuzzy clustering. In the proposed intra-cluster compactness measure, we compute the relative dispersion of data points in all the dimensions and try to minimize the maximum dispersion to increase the overall compactness within the clusters. In the proposed inter-cluster

separation measure based on the fuzzy set, we compute the distance between the centroids of fuzzy clusters and try to maximize the distance between the centroid of less separated fuzzy clusters to increase the overall separation among all pairs of fuzzy clusters. In the proposed inter-cluster overlap measure, we compute the degree of overlap between the fuzzy clusters. In this, we minimize the degree of overlap among the highly overlapped fuzzy clusters to reduce the overall overlapping among the fuzzy partitions. Hence, the optimal number of clusters determined by the proposed validity index for fuzzy clustering is expected to have high compactness, less degree of overlap, and greater separation between the fuzzy clusters.

The results show that the VI_{DSO} index outperforms over other indexes by correctly identifying the optimal number of clusters and the corresponding fuzzy partitions. Moreover, the optimal number of clusters determined by the VI_{DSO} index does not change with a change in values of m. Thus, we conclude that the VI_{DSO} index is considered as the reliable index and thus enhances the fuzzy clustering capability drastically.

Chapter 4

Methodologies for Selection of Optimal Parameters in Fuzzy Clustering

4.1 Introduction

As discussed in the previous chapter, for the selection of the optimal number of clusters (c), a novel cluster validity index is designed. There are a few more parameters on which clustering efficiency depends, i.e., the fuzzifier (m), and the initial cluster centers (V). A random selection of these parameters leads fuzzy clustering to trap into the problem of local optima. The fuzzifier controls the extent of sharing among fuzzy clusters [22]. It is a key parameter, which significantly affects the result of fuzzy clustering. Although, there have been many studies for the selection of m in fuzzy clustering [24, 25, 74, 82, 101, 102], Pal and Bezdek [74], suggested that an optimal value of m lies in the range of $[1.5, 2.5]$. But, there is still not one generalized form of criteria for selection of the optimal value of m in this range.

Another important factor in fuzzy clustering is the selection of the optimal number of clusters. In cluster analysis, determining the number of clusters plays an important role in obtaining practical and sound results. Many cluster validity indexes exist for the selection of the optimal number of clusters [76, 92–98] and some of these indexes also assist in selecting the optimal value of fuzzifier. However, these validity indexes do not take into consideration all the necessary

measures of cluster analysis. One more important factor in fuzzy clustering is the selection of initial cluster centers as it has a direct impact on the formation of final clusters. Many hybrid approaches [26,27,42,43,103,104,160] exist for the selection of initial cluster centers. In spite of the wide popularity of above-stated approaches and applications, initialization of parameters for fuzzy clustering is still a challenging task.

So, there is an immense need to find the appropriate value of m, c, and V by using the quantum computing to resolve the local convergence problem of fuzzy clustering. For this reason, we propose two algorithms, i.e., the QIE-FCM, and the EQIE-FCM. The proposed approaches utilize the quantum computing in fuzzy clustering to find the optimal value of parameters during the clustering process. The advantage of using the quantum computing is that it provides better characteristics of population diversity in comparison with other representations due to a linear superposition of states probabilistically. Thus, it provides a large search space for the selection of optimal solution by maintaining a proper balance between exploration and exploitation [107].

4.2 Proposed Work

This section presents details of two proposed algorithms, i.e., the QIE-FCM and the EQIE-FCM designed for finding the optimal parameters of fuzzy clustering. In both the proposed algorithms, i.e., QIE-FCM and EQIE-FCM the evolutionary quantum computing concept is utilized. The motivation behind using the evolutionary quantum computing concept is to evolve the parameters of fuzzy clustering. This concept uses a quantum bit in place of a classical bit to evolve the parameters. The evolutionary quantum computing concept is characterized by the representation of the individual, the evaluation function, and the population dynamics. The quantum bit Q is made up of a string of qubits and can represent a linear superposition of states in a search space probabilistically. Thus quantum bit representation has a better characteristic of population diversity (exploration) than other representations. Here, quantum rotational gate is also used as a variation operator to drive the individuals towards better solutions and eventually toward a single state. As the probability of each qubit approaches either 1 or 0 by the quantum rotational gate, the qubit individual

converges to a single state. Thus, it helps to achieve the exploitation. By this inherent mechanism, quantum evolutionary concept maintains a balance between exploration and exploitation.

In the proposed QIE-FCM algorithm, we utilize the concept of quantum computing [107] for evolving the parameter m of the fuzzy clustering in several generations ($g_{\max}$). In this, the obtained value of m in each generation (g) through the observation process is utilized for the execution of fuzzy clustering which helps to achieve the exploration. Within each generation, the fuzzy clustering is executed multiple times with the obtained value of m by varying the values of c. In this, the local and global fitness functions are designed by using a cluster validity index (VI_{DSO}) [161] which evaluates the fitness of every fuzzy partition obtained for each value of c. The minimum value of the fitness function indicates the optimal number of clusters corresponding to the value of m obtained in each generation. Then a different value of m is evolved in each generation by using the quantum rotational gate [109]. The update of the quantum bit of fuzzifier with the quantum rotational gate provides a proper exploitation during evolution. Thus, by comparing the fitness values of the local and global functions in each generation, a proper balance between exploration and exploitation is achieved. This comparison helps to shift the quantum bit in a proper direction where the optimal value of the parameter can be achieved. Hence, after several generations of evolution, the proposed algorithm gives the optimal number of clusters (c_{best}) and the best value of fuzzifier (m_{best}) from a large search space by maintaining a proper balance between exploration and exploitation. The limitation of this approach is a random selection of initial cluster centers. Due to this, the proposed QIE-FCM algorithm takes a higher number of iterations to determine the cluster center locations, which leads to a slow convergence problem.

Next, we propose an enhancement of the QIE-FCM algorithm, referred to as the EQIE-FCM algorithm. In this approach, we utilized the quantum computing [107] in fuzzy clustering to evolve the parameters, i.e., m and V from a large search space in $g_{\max}$ generations. In each generation, the obtained value of m and V (for each c) through the observation process helps to achieve the exploration which is utilized for the execution of fuzzy clustering. Within the generation, the fuzzy clustering process is executed multiple times with the

obtained value of m and V (for each m the V is varied for every value of c). In this, the local and global fitness functions are designed by using the cluster validity index VI_{DSO} [161], which is used to evaluate the fitness of every partition obtained for each value of c. The parameters m and V in each generation are evolved by updating quantum bits of these parameters using the quantum rotational gate given by Han and Kim [109]. The update of quantum bits of these parameters with the quantum rotational gate provides a proper exploitation during evolution. Thus, by comparing the global fitness function with local fitness function in each generation, a proper balance between exploration and exploitation is achieved. This comparison of fitness functions in each generation helps to shift the quantum bits in a proper direction where the optimal value can be achieved. After executing the EQIE-FCM algorithm for several generations, it gives the m_{best}, c_{best} and the best location of cluster centers (v_{best}) from a large search space by maintaining a proper balance between exploration and exploitation. The details of the same are presented next.

4.2.1 Representation of Fuzzifier and Initial Cluster Centers in Qubits

In this section, we have used an evolutionary principle of quantum computing to represent the fuzzifier and initial cluster centers in terms of a quantum bit. The preliminaries of quantum computing are already discussed in sections 2.4, The fuzzifier in each generation is represented in terms of a quantum bit defined as follows:

$$M^g = Q^g \tag{4.1}$$

where M^g is a quantum representation of fuzzifier for a generation and Q^g represents a quantum bit for a generation. Here, Q^g is represented in terms of only two-qubits which is defined as follows:

$$Q^g = (q_1^g | q_2^g) \tag{4.2}$$

where q_k^g denotes a k^{th} qubit for a generation, $k = 1, ..., h$, and $h = 2$, h represents the total number of qubits in a quantum bit. The Eq. (4.2) can be rewritten

using Eqs. (2.18) and (2.19) as follows:

$$Q^g = \langle \alpha_1^g | \alpha_2^g \rangle \tag{4.3}$$

where α_k^g is a complex number representing the probability of a k^{th} qubit for generation g, $k = 1, ..., h$, and $h = 2$. Here, a quantum bit is represented in terms of only two-qubits. This is because the best value of m has to be found within the range of $[1.5, 2.5]$ as suggested by Pal and Bezdek [74]. It is a single dimension real value that can be efficiently searched from four subspaces generated by the probabilistic representation of two-qubits of a quantum bit, which is already discussed in section 2.4.

For each value of m represented in terms of a quantum bit for generation g, the value of c is initialized in the range of $[c_{\min}, ..., c_{\max}]$. In general, a set of initial centers for c clusters is represented as follows:

$$V^g = \{v_1^g, v_2^g, ..., v_c^g\} \tag{4.4}$$

where V^g denotes a set of initial cluster centers for a generation, v_j^g denotes the j^{th} cluster center for a generation, and $j = 1, 2, ..., c$. $v_j^g \in R^d$ represents a j^{th} cluster center in a generation consists of d dimensions defined as follows:

$$v_j^g = \{v_{1j}^g, v_{2j}^g, ..., v_{dj}^g\} \tag{4.5}$$

where v_{fj}^g represents a f^{th} dimension of the j^{th} cluster center in a generation and $f = 1, 2, .., d$. As stated above, the parameter m contains a single dimension real value. Therefore, a quantum bit consists of only two qubits given in Eq. (4.2) is enough to represent the parameter m in a generation g as shown in Eq. (4.1). Here, a set V^g is given in Eq. (4.4) consist of c cluster centers and each j^{th} cluster center for a generation consist of d-dimensions as shown in Eq. (4.5). Thus, each j^{th} cluster center in f^{th} dimension in a generation, i.e., v_{fj}^g contains a single dimension real value. Therefore, a quantum bit consists of two-qubits are enough to represent a j^{th} cluster center in f^{th} dimension where $f = 1, 2, ..., d$. Hence, multiple quantum bits are required to represent each j^{th} cluster center in a generation, i.e., v_j^g and a set V^g. However, the j^{th} cluster center in f^{th} dimension for a generation is represented in terms of a quantum bit, let us

assume a different notation for this denoted as $(v^{''})^g_{fj}$ and it is defined as follows:

$$(v^{''})^g_{fj} = Q^g_{fj} \tag{4.6}$$

where $(v^{''})^g_{fj}$ is a quantum representation of the j^{th} cluster center in f^{th} dimension for a generation, and Q^g_{fj} is a quantum bit of the j^{th} cluster center in f^{th} dimension for a generation. Here, Q^g_{fj} consists of only two-qubits, and it is represented as follows:

$$Q^g_{fj} = (q^g_{fj})_1|(q^g_{fj})_2 \tag{4.7}$$

As discussed above, the formulation of Eq. (4.3), similarly the Eq. (4.7) can be rewritten using Eqs. (2.18) and (2.19) and is redefined as follows:

$$Q^g_{fj} = \langle(\alpha^g_{fj})_1|(\alpha^g_{fj})_2\rangle \tag{4.8}$$

where $(\alpha^g_{fj})_k$ is a complex number representing the probability of a k^{th} qubit of j^{th} cluster center in f^{th} dimension for a generation, such that $k = 1, ..., h$, and $h = 2$. Here, a quantum bit Q^g_{fj} is represented in terms of only two-qubits. This is because j^{th} cluster center in f^{th} dimension represents a single real value which can be sufficiently found from four subspaces obtained by the two-qubits representation of a quantum bit as discussed in section 2.4.

It is important to notice that both the proposed algorithms, i.e., the QIE-FCM and the EQIE-FCM execute on a classical computer. Therefore, it is required that a quantum bit of M^g and $(v^{''})^g_{fj}$ has to be converted into a real-coded value. This conversion is carried out with the help of observation process that helps in achieving the exploration [110]. The preliminaries of observation process are given in section 2.4, and detailed description is showing the conversion of a quantum bit into a real-coded value is presented next.

4.2.2 Observation Process

In general, we present the observation process showing the conversion of a quantum bit Q^g into real-coded value $(Q^{'})^g$. The same procedure is applicable for the conversion of a quantum bit of M^g and $(v^{''})^g_{fj}$ into real-coded value represented as m^g and v^g_{fj}, respectively. The observation process starts with the selection of a random number vector for a generation, i.e., ρ^g, where $\rho^g = [\rho^g_1 \rho^g_2]$. The

random number vector selected corresponds to a quantum bit $Q^g = (q_1^g|q_2^g)$ or $Q^g = \langle \alpha_1^g|\alpha_2^g\rangle$ given in Eqs. (4.2) and (4.3) or corresponds to a quantum bit $Q_{fj}^g = (q_{fj}^g)_1|(q_{fj}^g)_2$ or $Q_{fj}^g = \langle(\alpha_{fj}^g)_1|(\alpha_{fj}^g)_2\rangle$ given in Eqs. (4.7) and (4.8). Then, a further mapping is done by generating a corresponding bit in a binary vector for a generation, i.e., b^g where $b^g = [b_1^g, b_2^g]$ and the Gaussian random number generator (grg) with a mean value μ_k^g and variance σ_k^g for a generation represented as $grg(\mu_k^g, \sigma_k^g)$, where $k = 1, .., h$, and $h = 2$. Using the random number vector and a quantum bit, the binary vector b^g is generated as follows:

$$\text{if } (\rho_k^g \leq (\alpha_k^g)^2) \text{ then } b_k^g = 1 \text{ else } b_k^g = 0.$$

Now, with the help of binary vector and the Gaussian random number generator, the real-coded value is selected using a formula $bin2dec(b^g) + 1$.

The observation process is presented in terms of pseudo-code showing the conversion of a quantum bit into a real-coded value for a generation (g) represented as $(Q^{'})^g$. This conversion is presented as follows:

Observation Process()

Begin

Step-1: Initialize a quantum bit Q^g, a random number vector ρ^g and $link = 0$.

for $k := 1$ to 2 **step** 1 **do**

$Q^g = \alpha_k^g; \quad 0 \leq \alpha_k^g \leq 1$

$\rho_k^g = rand;$

end for

Step-2: **for** $k := 1$ to 2 **step** 1 **do**

if $\rho_k^g \leq (\alpha_k^g)^2$

$b_k^g = 1;$

else

$b_k^g = 0;$

end if

end for

Step-3: $link = bin2dec(b^g) + 1$

if $link \sim= 0$

$k = link$;

$(Q^{'})^g = grg(\mu_k^g, \sigma_k^g)$;

end if

End

return $(Q^{'})^g$

Once, the observation process is completed, a real-coded value for the fuzzifier m^g is represented as follows:

$$m^g = (Q^{'})^g \tag{4.9}$$

Similarly, the real-coded value of cluster centers v_{fj}^g is represented as follows:

$$v_{fj}^g = (Q^{'})^g \tag{4.10}$$

where $f = 1, 2, .., d$ and $j = 1, 2, .., c$ such that the real-coded values of Eqs. (4.4) and (4.5) are obtained using Eq. (4.10).

Now, the real-coded value of m^g is used in both the proposed algorithms, i.e., the QIE-FCM and the EQIE-FCM. Similarly, the real-coded value of cluster centers v_{fj}^g is used in the EQIE-FCM algorithm. In both the proposed algorithms, a different value of the fuzzifier and initial cluster centers are evolved in each generation by using the concept of the quantum rotational gate [109], which is presented next.

4.2.3 Working of Quantum Rotational Gate

The quantum rotational gate [109] is used to update the qubits of fuzzifier and cluster centers in each generation so that a proper exploitation among the solutions can be achieved. The update of qubits with the quantum rotational gate is defined as follows:

$$\alpha_k^{g+1} = [(\alpha_k^g * cos(\Delta\theta)) - (\sqrt{1 - (\alpha_k^g)^2} * sin(\Delta\theta))] \tag{4.11}$$

where rotational angle or angular displacement ($\Delta\theta$) provides a proper angle for rotating qubits so that we can get proper exploitation for selecting the appropriate value of a new quantum bit. This concept of the quantum rotational gate is used in the proposed algorithms to update qubits in each generation. Thus,

the proposed algorithms, i.e., the QIE-FCM and the EQIE-FCM is detailed subsequently.

4.2.4 Quantum-inspired Evolutionary Fuzzy Clustering Algorithm

This section discusses the use of quantum computing concept in the proposed QIE-FCM algorithm. In this algorithm, the value of fuzzifier is evolved in each generation by using the evolutionary principle of quantum computing. In the proposed algorithm, qubits of the fuzzifier in each generation are evolved by updating it using a concept of the quantum rotational gate [109] as presented in Eq. (4.11). In this equation, $\Delta\theta$ is calculated on the basis of two fitness functions, i.e., global function ($F_{Gbest}(m_{best}, c_{best})$) and local function ($F^g_{Lbest}(m^g, c)$) and it is also selected on the basis of conditions presented in Table 4.1. However, formulations of these fitness functions are designed by using the VI_{DSO} index [161]. The formulation of local fitness function (($F^g_{Lbest}(m^g, c)$) is given in Eq. (4.16) whereas the global fitness function $F_{Gbest}(m_{best}, c_{best})$ is determined as follows:

$$F_{Gbest}(m_{best}, c_{best}) = \min(F_{Gbest}(m_{best}, c_{best}), F^g_{Lbest}(m^g, c)) \tag{4.12}$$

where, $F^g_{Lbest}(m^g, c)$ is a minimum value of the fitness function determined corresponding to m^g by varying the value of $c = [c_{\min}, ..., c_{\max}]$. The global fitness function, i.e., $F_{Gbest}(m_{best}, c_{best})$ is initially assumed to be infinity (∞) and is computed by storing the best value of local fitness function obtained among all the generations. Thus, the global fitness function generates the best value of fuzziness parameter and the optimal number of clusters denoted as m_{best} and c_{best}, respectively.

In this, the value of $F_{Gbest}(m_{best}, c_{best})$ and $F^g_{Lbest}(m^g, c)$ are derived corresponding to the value of m^g, which is generated through a quantum bit (M^g). Also, the observation process shows that each qubit α^g_k is associated with a binary bit b^g_k. Therefore a mapping is done between $F_{Gbest}(m_{best}, c_{best})$, $F^g_{Lbest}(m^g, c)$, and b^g_k to update each qubit. The values of $F_{Gbest}(m_{best}, c_{best})$ and $F^g_{Lbest}(m^g, c)$ is obtained corresponding to b^{global}_k and b^g_k where b^{global}_k and b^g_k is a binary bit corresponding to $F_{Gbest}(m_{best}, c_{best})$ and $F^g_{Lbest}(m^g, c)$, respectively. If

Table 4.1: Parameters for qubits updation of the QIE-FCM algorithm.

b_k^g	b_k^{global}	$F_{Gbest}(m_{best}, c_{best}) > F_{Lbest}^g(m^g, c)$	$\Delta\theta$
0	*0*	*false*	*0*
0	*0*	*true*	*0*
0	*1*	*false*	$-0.03 * \Pi$
0	*1*	*true*	*0*
1	*0*	*false*	*0*
1	*0*	*true*	$0.03 * \Pi$
1	*1*	*false*	*0*
1	*1*	*true*	*0*

the value of $F_{Lbest}^g(m^g, c)$ obtained in the current generation is worse than the value of $F_{Gbest}(m_{best}, c_{best})$ obtained in the previous generation, and state of b_k^g is zero in the current generation, and b_k^{global} is one, then decreasing the probability of α_k^g to zero may produce the worst result. Therefore, to update α_k^g, it is required that $\Delta\theta$ must be negative. On the contrary, if the value of $F_{Lbest}^g(m^g, c)$ obtained in the current generation is better than the value of $F_{Gbest}(m_{best}, c_{best})$ obtained in the previous generation, and state of b_k^g is one in the current iteration and b_k^{global} is zero, then increasing the probability of α_k^g to one may produce the worst result. Therefore, to update α_k^g, $\Delta\theta$ must be positive. In other cases, $\Delta\theta$ remains zero. The value of $\Delta\theta$ must be selected in such a way so that it takes a minimum number of iterations to cover a maximum number of values of α_k^g in the range of (0, 1). Hence, according to Han and Kim [109], $\Delta\theta$ must be initialized between $[0.01 \times \pi, 0.05 \times \pi]$.

From preventing the quantum bit α_k^g from attaining values 0 or 1, following constraints are applied:

$$\alpha_k^g = \begin{cases} \sqrt{\gamma}, & if \quad \alpha_k^g < \sqrt{\gamma} \\ \alpha_k^g & if \quad \sqrt{\gamma} \leq \alpha_k^g \leq \sqrt{1-\gamma} \\ \sqrt{1-\gamma} & if \quad \alpha_k^g > \sqrt{1-\gamma} \end{cases} \tag{4.13}$$

where the limiting parameter γ is assigned a very small value (approximately approaching to zero) so that it can cover maximum values in the range of (0, 1).

Next, we present the working of proposed QIE-FCM algorithm which is executed for $g_{\max}$ generations where $g_{\max}$ is set to 100 to find the best value of fuzzifier and the optimal number of clusters. As suggested by Pal and Bezdek

[74], an appropriate value of m lies in the interval of $[1.5, 2.5]$. Therefore, the QIE-FCM algorithm is executed for only 100 generations because the values lying within the interval $[1.5, 2.5]$ are sufficiently explored in 100 generations. It means that a significant exploration and exploitation is achieved within 100 generations. If the QIE-FCM algorithm is executed for a higher number of generations, i.e. greater than 100, then it is not changing the value of fuzzifier at all and unnecessarily increasing the computational overhead. The QIE-FCM algorithm is presented next in the form of ***Algorithm 4.1***.

Algorithm 4.1 Algorithm for Quantum-inspired Evolutionary Fuzzy Clustering

1: **Input** $X = \{x_1, x_2, ..., x_n\}$ and use Table 4.2 to initialize the parameters.
2: **Output** $F_{Gbest}(m_{best}, c_{best})$.
3: **Begin**
4: **Set** the maximum number of generations $g_{\max}$ to 100, and the current generation number g is initialized as 1.
5: **while** $g \leq g_{\max}$ **do**
6: **Initialize** m for a generation in the form of a quantum bit using Eq. (4.1).
7: **Call observation process**: To obtain a real-coded value (m^g) corresponding to a quantum value M^g using Eq. (4.9).
8: **Initialize** parameters related to the fuzzy clustering using Table 2.1.
9: **for** $c := c_{\min}$ **to** $c_{\max}$ **step** 1 **do**
10: **Execute** the fuzzy clustering stated in ***Algorithm 2.2*** for each value of c.
11: **end for**
12: **Compute** the objective function $VI_{DSO}(c, U^g)$ [161] using Eq. (3.23) to evaluate the fitness of obtained partitions for each value of c corresponding to m^g.
13: **Compute** the summation of $VI_{DSO}(c, U^g)$ as follows:

$$VI_{DSO}^{sum}(c, U^g) = \sum_{c=c_{\min}}^{c_{\max}} VI_{DSO}(c, U^g) \tag{4.14}$$

14: **for** $c := c_{\min}$ **to** $c_{\max}$ **step** 1 **do**
15: **Compute** the normalized value of $VI_{DSO}(c, U^g)$ for each value of c as follows:

$$VI_{DSO}^{Normalized}(c, U^g) = \frac{VI_{DSO}(c, U^g)}{VI_{DSO}^{sum}(c, U^g)} \tag{4.15}$$

16: **end for**
17: **Compute** the local fitness value for a generation, i.e., $F_{Lbest}^g(m^g, c)$ as follows:

$$F_{Lbest}^g(m^g, c) = \min_{c_{\min} \leq c \leq c_{\max}} [VI_{DSO}^{Normalized}(c, U^g)] \tag{4.16}$$

18: **Compute** $F_{Gbest}(m_{best}, c_{best})$ using Eq. (4.12).
19: **Update** quantum bit (M^g) by using Table 4.1, Eqs. (4.11) and (4.13).

Algorithm 4.1 (Continued)

20: **Update** $g = g + 1$.
21: **end while**
22: **End**

Table 4.2: Specification of parameters of QIE-FCM algorithm.

Parameters	Description	Values
σ	Variance	0.6
$\Delta\theta$	Rotational angle	$0.03 \times \pi$
γ	Limiting parameter	0.01
$g_{\max}$	Maximum number of generations	100
$F_{Gbest}(m_{best}, c_{best})$	Global fitness function	∞

One of the major limitation of the proposed QIE-FCM algorithm is that in this, the location of initial cluster centers is chosen randomly. Due to this, the QIE-FCM algorithm converges slowly by taking a higher number of iterations to determine the cluster center locations. Therefore, to overcome the drawback of the QIE-FCM algorithm, we proposed another algorithm named as Enhanced Quantum-inspired Evolutionary Fuzzy Clustering Algorithm (EQIE-FCM), which is discussed next.

4.2.5 Enhanced Quantum-inspired Evolutionary Fuzzy Clustering Algorithm

In the proposed EQIE-FCM algorithm, we apply the evolutionary quantum computing to evolve two parameters, i.e., the fuzzifier and initial cluster centers. In this algorithm, the qubits of fuzzifier and initial cluster centers are evolved in each generation. To do this, the quantum rotational gate [109] is used which plays an important role in quantum inspired approaches. The quantum bits of these parameters are updated using Eqs. (4.11) and (4.13). In these equations, the rotational angle ($\Delta\theta$) is calculated by using the conditions presented in Table 4.3.

Next, we present the working of proposed EQIE-FCM algorithm, which is also executed for $g_{\max}$ generations, similar to the proposed QIE-FCM algorithm. The step-wise procedure of the EQIE-FCM algorithm for the selection of the optimal number of clusters (c_{best}), the best value of fuzzifier (m_{best}), and the best location of initial cluster centers (v_{best}) is presented next in the form of ***Algorithm 4.2***.

Algorithm 4.2 Algorithm for Enhanced Quantum-inspired Evolutionary Fuzzy Clustering

Input $X = \{x_1, x_2, ..., x_n\}$ and use Table 4.4 to initialize the parameters.
Output $F_{Gbest}(m_{best}, c_{best})$.
Begin
Current generation g is initialized as 1 and set the maximum number of generations g_{max} to 100.
while $g \leq g_{max}$ **do**
  Initialize m for a generation in terms of quantum bits using Eq. (4.1).
  Call observation process: To obtain the real-coded value m_g corresponding to the quantum value M^g using Eq. (4.9).
  Initialize parameters related to fuzzy clustering using Table 2.1.
  for $c := c_{\min}$ **to** $c_{\max}$ **step** 1 **do**
    Initialize the cluster centers in terms of a quantum bit using Eq. (4.6).
    Call observation process: To obtain the real-coded value v_{fj}^g corresponding to the quantum value $(v^{''})_{fj}^g$ using Eq. (4.10).
    Execute the fuzzy clustering stated in ***Algorithm 2.2*** for each value of c.
  end for
  Compute the VI_{DSO} index [161] using Eq. (3.23) which is used as the objective function $VI_{DSO}(c, U^g)$ in this algorithm to evaluate the fitness of obtained partitions for all the values of c corresponding to m^g.
  Compute the summation of $VI_{DSO}(c, U^g)$ using Eq. (4.14).
  for $c := c_{\min}$ **to** $c_{\max}$ **step** 1 **do**
    Compute the normalized value of $VI_{DSO}(c, U_g)$ for each value of c using Eq. (4.15).
    Store the fitness of fuzzy partition corresponding to each c in F_c^g defined as follows:

$$F_c^g = VI_{DSO}^{Normalized}(c, U^g) \tag{4.17}$$

    if $(F_c^g \leq F_c^{gbest})$ **then**
      $F_c^{gbest} = F_c^g$
      $v_{best} = V^g$
      Update the quantum bit of $(v^{''})_{fj}^g$ by using Table 4.3, Eqs. (4.11) and (4.13).
    else
      $F_c^{gbest} = F_c^{gbest}$
      $v_{best} = v_{best}$
      Update the quantum bit of $(v^{''})_{fj}^g$ by using Table 4.3, Eqs. (4.11) and (4.13).
    end if
  end for
  Compute the local fitness value for generation (g), i.e., $F_{Lbest}^g(m^g, c)$ using Eq. (4.16).
  Compute $F_{Gbest}(m_{best}, c_{best})$ using Eq. (4.12) to identify the m_{best}, c_{best} and the v_{best} from the overall generations.

Algorithm 4.2 (Continued)

31: **Update** the quantum bit of (M^g) by using Table 4.3, Eqs. (4.11) and (4.13).
32: **Update** $g = g + 1$.
33: **end while**
34: **Return** m_{best}, c_{best} and v_{best}.
35: **End**

Table 4.3: Parameters for qubits updation of the EQIE-FCM algorithm.

b_k^g	b_k^{global}	$F_{Gbest}(m_{best}, c_{best}) > F^g_{Lbest}(m^g, c)$ or $F_c^{gbest} > F_c^g$	$\Delta\theta$
0	*0*	*false*	*0*
0	*0*	*true*	*0*
0	*1*	*false*	$-0.03 * \Pi$
0	*1*	*true*	*0*
1	*0*	*false*	*0*
1	*0*	*true*	$0.03 * \Pi$
1	*1*	*false*	*0*
1	*1*	*true*	*0*

4.3 Experimental Evaluation

In this section, we present the datasets and the experimental settings used in experimental evaluations. We compare the performance of proposed QIE-FCM algorithm with standard validity indexes, i.e., Rezaee et al. index (V_{CWB}) [93] and Kim and Lee index (v_{OS}) [95]. This is because the proposed QIE-FCM algorithm uses VI_{DSO} index in the fitness function. Therefore, to compare the fitness value, it is good to report the comparison of the proposed QIE-FCM algorithm with the other cluster validity indexes which also evaluates the fitness value using an almost similar kind of measures. Thus, other compared indexes also evaluate the fitness value based on the combination of compactness, separation, and overlap measures like our proposed VI_{DSO} index. Thus, for reporting these comparisons, other indexes are also executed for 100 independent runs using the similar value of fuzzifier which is taken in each generation of the QIE-FCM algorithm. Furthermore, the comparison of the one more proposed EQIE-FCM algorithm is done with the proposed QIE-FCM algorithm [162]. Furthermore, the comparisons of both the proposed algorithms, i.e., QIE-FCM and EQIE-FCM is done with other evolutionary clustering algorithms [42, 43, 104].

Table 4.4: Specification of parameters of EQIE-FCM algorithm.

Parameters	Description	Values
σ	Variance	0.6
$\Delta\theta$	Rotational angle	$0.03 \times \pi$
γ	Limiting parameter	0.01
$g_{\max}$	Maximum number of generations	100
v_{best}	Best location of cluster centers	ϕ
F_c^{gbest}	Global best fitness for c^{th} cluster	∞
$F_{Gbest}(m_{best}, c_{best})$	Global best fitness function	∞

4.3.1 Datasets and Experimental Settings

The effectiveness of QIE-FCM algorithm is evaluated on six well-known datasets, i.e., IRIS, WINE, GLASS, VEHICLE, SEED, and BUPA. Also, the performance of the EQIE-FCM algorithm is evaluated on all these datasets along with the PID dataset. These datasets are taken from the UCI Machine Learning repository [154]. The detailed description of these datasets is given in Appendix A. All the experiments are carried out on Intel(R) Xeon(R) E5-1607 v3 @ 3.10GHz x 4, workstation PC with 64 GB of memory and running on the Windows 7 Professional operating systems. The implementation is done in the MATLAB computing environment and executed on MATLAB version R2014a. The setting of various parameters for the QIE-FCM algorithm and the EQIE-FCM algorithm is presented in Tables 4.2 and 4.4.

4.3.2 Experimental Results and Discussion

In this section, first the results of a comparative analysis for the QIE-FCM algorithm is presented, and then we show the result of comparative analysis of the EQIE-FCM algorithm.

Comparative result of QIE-FCM algorithm

In this section, the performance of proposed QIE-FCM algorithm is evaluated in comparison with two well-known fuzzy based clustering indexes [93, 95]. In addition to this, the superiority of QIE-FCM algorithm is also investigated by comparing it with three evolutionary clustering algorithms [42, 43, 104]. The efficacy of QIE-FCM algorithm is judged in terms of the best value of fitness function, identification of the best value of fuzzifier, and determination of the

optimal number of clusters, which is discussed next.

Best fitness value analysis

In Figure 4.1, for each dataset, the best fitness value of the proposed QIE-FCM algorithm is reported in comparison with the V_{CWB} [93] and v_{OS} [95] indexes on different values of the fuzzifier obtained in 100 generations. The reason for comparing the proposed QIE-FCM algorithm with the V_{CWB} and v_{OS} indexes is that in the QIE-FCM algorithm, the fitness functions are calculated by using the VI_{DSO} index [161] which is based on three important measures such as compactness, separation, and overlap and thus evaluates the fitness of obtained fuzzy partitions. The small value of the VI_{DSO} index indicates the best fitness value which leads to better fuzzy partitions. The compared indexes are also based on the combination of these measures and evaluate the fitness. So to compare our proposed approach in terms of appropriateness of proposed VI_{DSO} index, we compared with these two indexes in terms of fitness value.

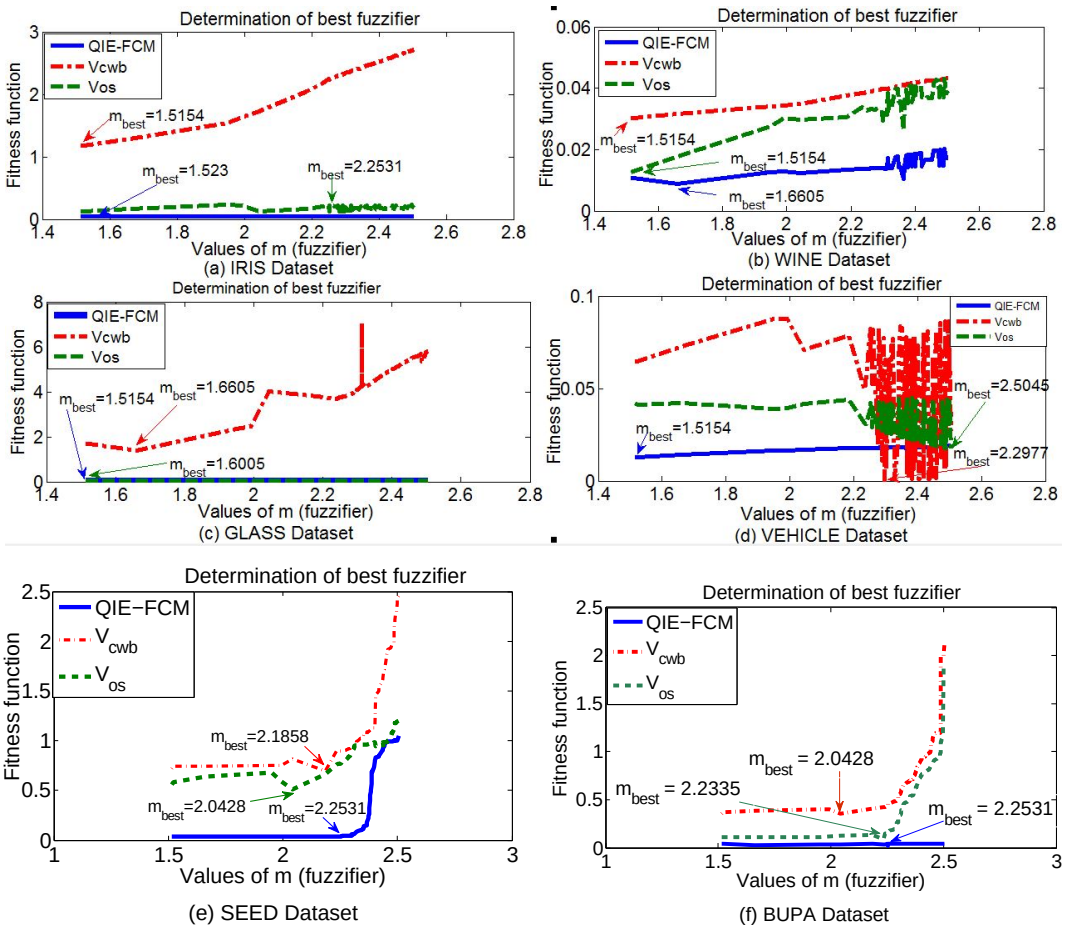


Figure 4.1: Comparison of the QIE-FCM algorithm with other cluster validity indexes indicating the best value of fuzzifier for different datasets.

Since our approach is working on evolutionary quantum computing principle, therefore it is giving better fitness value as compared these two indexes. It is observed from Figure 4.1, the best fitness value achieved by the QIE-FCM algorithm on IRIS dataset is at $m_{best} = 1.523$, that is comparatively 29.45 times and 3.23 times lesser than the best fitness value obtained by the V_{CWB} at $m_{best} = 1.5154$ and v_{OS} at $m_{best} = 2.2531$. On WINE dataset, the QIE-FCM algorithm achieved best fitness value at $m_{best} = 1.6605$ that is in comparison 3.44 times and 1.43 times lesser than the best fitness value attained by the V_{CWB} and v_{OS} at $m_{best} = 1.5154$. Furthermore, for GLASS dataset, the QIE-FCM algorithm achieves the best fitness value at $m_{best} = 1.5154$, that is comparatively 96 times and 3.9 times lesser the best fitness value achieved by the V_{CWB} at $m_{best} = 1.6605$ and v_{OS} at $m_{best} = 1.6605$. On VEHICLE dataset, the best fitness value attains by the QIE-FCM algorithm is at $m_{best} = 1.5154$, that is in comparison 1.33 times lesser than the best fitness value obtained by the v_{OS} at $m_{best} = 2.5045$ and 64.47 times higher than the best fitness value obtained by the V_{CWB} at $m_{best} = 2.2977$. On SEED dataset, the QIE-FCM achieves the best fitness value at $m_{best} = 2.2531$ which is in comparison 20.02 times and 14.74 times lesser than the best fitness value obtained by the V_{CWB} at $m_{best} = 2.1858$ and v_{OS} at $m_{best} = 2.0428$, respectively. On BUPA dataset, the best fitness value achieved by the QIE-FCM is at $m_{best} = 2.2531$ which is comparatively 16.41 times and 4.22 times lesser than the fitness value attained by the V_{CWB} at $m_{best} = 2.0428$ and v_{OS} at $m_{best} = 2.2335$, respectively. It can be seen from the above-reported results that, the QIE-FCM algorithm achieves the better fitness value in comparison with the V_{CWB} and v_{OS} indexes for IRIS, WINE, GLASS, SEED, and BUPA datasets, respectively. But, in the case of VEHICLE dataset, the QIE-FCM algorithm achieves the better fitness value only in comparison with the v_{OS} index.

Identification of number of clusters versus fuzzifier

As suggested by Pal and Bezdek [74], the fuzzy clustering achieves the best results for $m \in [1.5, 2.5]$. Furthermore, the fuzzy based approaches or the validity indexes are considered reliable when the number of clusters identified by these approaches is insensitive to change in values of fuzzifier. Therefore, in Figure 4.2 for each dataset, the number of clusters identified by the QIE-FCM algorithm

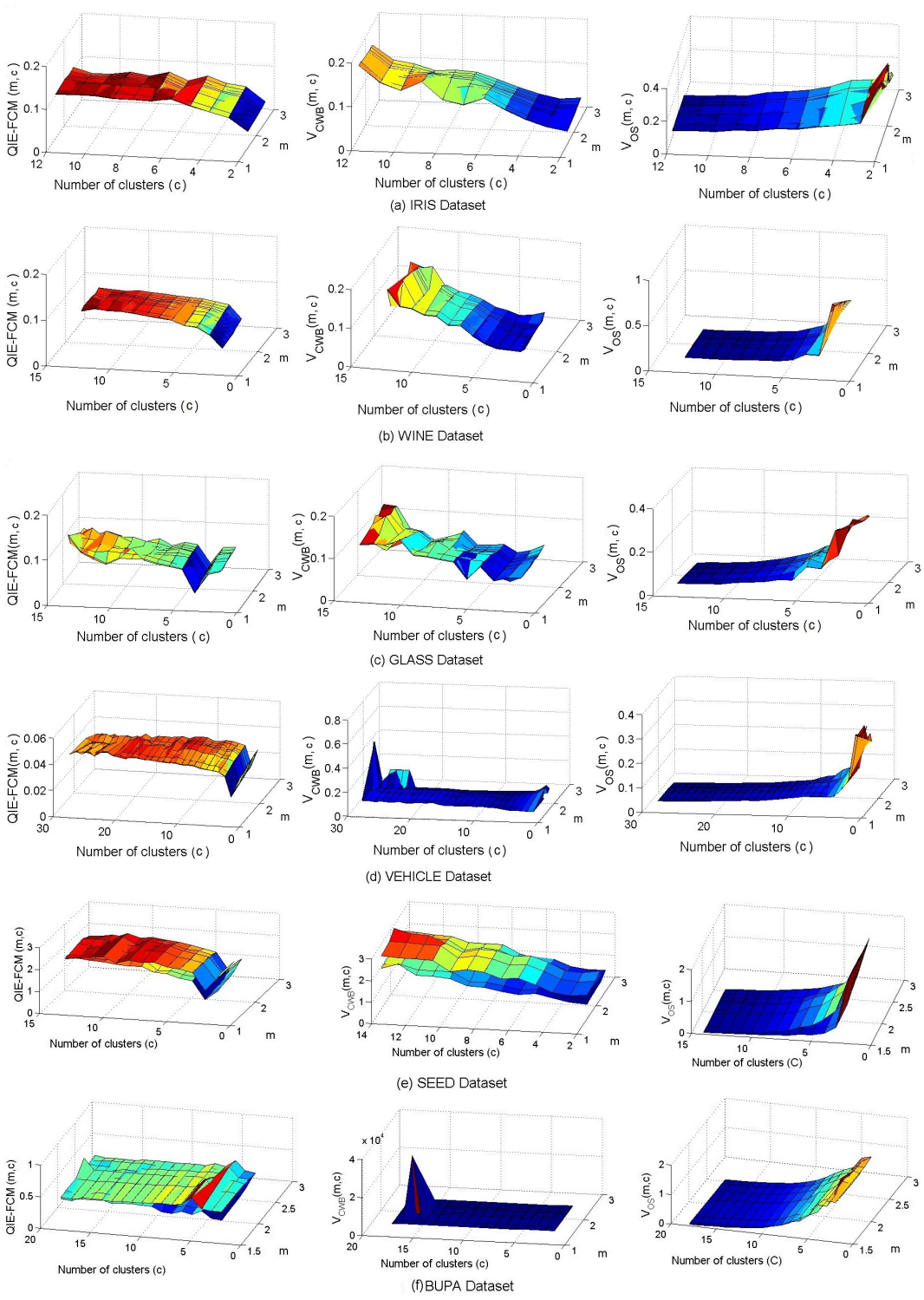


Figure 4.2: Comparison of the QIE-FCM algorithm with other cluster validity indexes in terms of the number of clusters for different datasets.

and the V_{CWB} [93] and v_{OS} [95] indexes on ten different values of $m \in [1.5, 2.5]$ by varying c from $[c_{\min}, ..., c_{\max}]$; $c_{\min} = 2$; $c_{\max} = \sqrt{n}$ is reported. The performance of the QIE-FCM algorithm is evaluated in comparison with the V_{CWB} and v_{OS} indexes, respectively. This is because the QIE-FCM algorithm makes use of the VI_{DSO} index [161] to evaluate the fitness of fuzzy partitions, therefore,

it is necessary to evaluate the performance of the QIE-FCM algorithm in comparison with the V_{CWB} and v_{OS} indexes. In case of above-reported datasets, for all values of m, the QIE-FCM algorithm always correctly identified the optimal number of clusters which is similar to the number of clusters can be seen in the distribution of data shown in Figure 3.4. Also, the number of clusters obtained by the QIE-FCM algorithm is constant and does not change with a change in m values. Therefore, the proposed QIE-FCM algorithm is considered reliable because it always correctly identifies the number of clusters which are insensitive to change in m. However, in the case of all six datasets for some values of m, the V_{CWB} index can correctly identify the number of clusters. Also, the number of clusters identified by the V_{CWB} index changes with a change in values of m therefore, it is not considered reliable in validating the fuzzy partitions and in predicting the number of clusters correctly. Similarly, for the above-stated dataset, the v_{OS} index is unable to identify the number of clusters correctly for all the values of m. This is because v_{OS} index monotonically decreases with increase in c value, which makes it hard to detect the optimal number of clusters. Thus, the optimal number of clusters cannot be rated based on the v_{OS} index and therefore it is not considered reliable in identifying the number of clusters correctly with respect to m.

We can see from the results reported in Table 4.5 that, for all the six datasets, the QIE-FCM algorithm always correctly identifies the optimal number of clusters. It also identifies the best value of fuzzifier corresponding to the best value of fitness function. In contrast, only on the IRIS, WINE, and SEED datasets, the V_{CWB} index [93] can correctly identify the optimal number of clusters. For the above-reported datasets, the v_{OS} index [95] is unable to identify the optimal number of clusters correctly. Furthermore, for all six datasets, the V_{CWB} and v_{OS} are unable to determine the best value of fuzzifier as a minimum value of the fitness function achieved by these indexes is much higher than the fitness value achieved by the QIE-FCM algorithm. Hence, it can be inferred from the above-reported results that, the QIE-FCM algorithm outperforms over the V_{CWB} and v_{OS} indexes in terms of various parameters. The value of fitness function achieved by the QIE-FCM algorithm is comparatively much lesser than other approaches. In addition to this, the above-reported results show that for all six benchmark datasets, corresponding to different values of fuzzifier obtained

Table 4.5: Comparison of QIE-FCM algorithm with other indexes in terms of the best value of fuzzifier and the optimal value of number of clusters.

Datasets	QIE-FCM				V_{CWB}				v_{OS}			
	m_{best}	c_{best}	fitness value	SD	m_{best}	c_{best}	fitness value	SD	m_{best}	c_{best}	fitness value	SD
IRIS	1.5230	2	0.0401	0.0985	1.5154	2	1.1784	1.4503	2.2531	12	0.1737	1.2922
WINE	1.6605	2	0.0108	0.0560	1.5154	2	0.0316	1.5768	1.5154	11	0.0127	2.0983
GLASS	1.5154	4	0.0143	0.0231	1.6605	3	1.3704	1.7483	1.6005	14	0.0612	1.3838
VEHICLE	1.5154	3	0.0129	0.0456	2.2977	16	0.0002	2.4738	2.5045	29	0.0172	1.3728
SEED	2.2531	3	0.03472	0.01610	2.1858	3	0.69521	2.0972	2.0428	14	0.5121	1.0982
BUPA	2.2531	2	0.021423	0.08362	2.0428	5	0.35167	1.3832	2.2335	18	0.09056	1.0737

in 100 generations, the QIE-FCM algorithm always correctly identified the optimal number of clusters which are insensitive to change in m values. This shows that the QIE-FCM algorithm is reliable over other existing approaches [93, 95].

Comparison with other methods

The proposed method is also compared with three different evolutionary clustering algorithms, i.e., the QM-FCM [43], the RQECA [104], and the FCMVGA [42], in terms of the optimal number of clusters and the best fitness value. In the proposed QIE-FCM approach, the fitness functions are formulated by using the VI_{DSO} index [161], which jointly incorporates all three measures, i.e., the intra-cluster compactness, the inter-cluster separation, and the inter-cluster overlap to evaluate the fitness in comparison with other approaches. Therefore, it is observed from the results reported in Table 4.6 that a minimum value of the fitness function achieved by the proposed approach is comparatively much lesser than the value of fitness function achieved by the other compared approaches. In addition to this, the proposed approach can identify the optimal number of clusters for the above-stated datasets in comparison with other approaches. It is important to highlight that proposed approach also identifies the best value of the fuzzifier corresponding to a minimum value of fitness function for these datasets. In contrast, the authors of comparative approaches are unable to ad-

Table 4.6: Performance comparison with evolutionary clustering algorithms.

Datasets	QIE-FCM			FCMVGA [42]			QM-FCM [43]			RQECA [104]		
	c_{best}	Best fitness value	SD	c_{best}	Best fitness value	SD	c_{best}	Best fitness value	SD	c_{best}	Best fitness value	SD
IRIS	2	0.0401	0.0985	6	4.8024	2.0938	4	0.3866	1.0283	5	0.7587	1.0922
WINE	2	0.0108	0.0560	12	4.03309	1.9739	5	0.0724	1.0982	7	0.4865	1.0083
GLASS	4	0.0143	0.0231	2	36.0576	2.0973	3	0.4201	1.9822	14	1.8975	1.0022
VEHICLE	3	0.0129	0.0456	5	48.0980	2.3473	5	1.4289	1.7362	25	4.5610	1.0237
SEED	3	0.03472	0.01610	4	8.7352	1.0983	6	2.5672	1.7382	12	1.6983	1.7282
BUPA	2	0.021423	0.08362	7	10.2568	1.0092	7	1.8956	2.3622	16	3.5689	1.0282

dress the issue of identifying best value fuzzifier for these datasets. Hence, the above-reported results in Table 4.6 quantify the effectiveness of the proposed approach over other evolutionary clustering approaches.

Comparative result of EQIE-FCM algorithm

In this section, we present results of the EQIE-FCM algorithm on seven benchmark datasets, i.e., IRIS, WINE, GLASS, VEHICLE, SEED, BUPA, and PID. The scatter plot of first six datasets are presented in Figure 3.4, and the scatter plot of PID dataset is shown in Figure 4.3. The results are evaluated in comparison with the QIE-FCM algorithm in terms of the identification of the best value of fuzzifier, the best location of cluster centers, and the number of iterations. In addition to this, comparative results are reported with other evolutionary clustering algorithms. The details of this are presented next.

Evaluation of best fitness value and fuzzifier parameter

The best value of fuzzifier and the fitness function achieved by the EQIE-FCM algorithm in comparison with the QIE-FCM algorithm for all the seven benchmark datasets are presented in Figure 4.4. The comparative results are reported on different values of the fuzzifier obtained in 100 generations. In both the algorithms, the fitness functions are formulated by using the VI_{DSO} [161] index. The fitness functions formulated in these algorithms are used to evaluate the fitness of obtained fuzzy partitions. The small value of VI_{DSO} index [161] indicates the minimum value of fitness function and thus represents the better fuzzy partitions. In this figure for IRIS dataset, the minimum value of the fitness function achieved by the EQIE-FCM algorithm is at $m_{best} = 1.523$ which is in

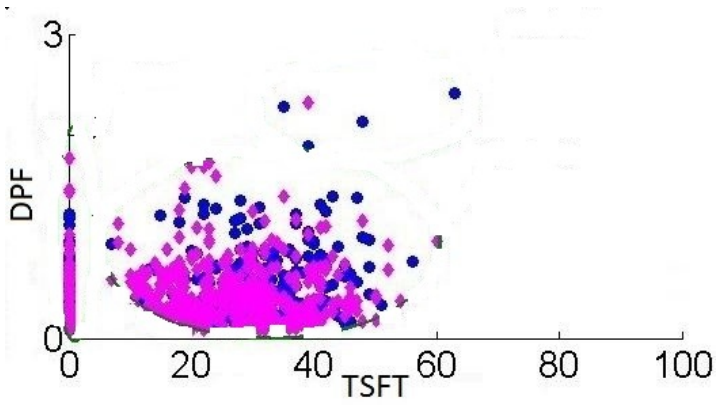


Figure 4.3: Scatter plot of Pima Indians Diabetes data.

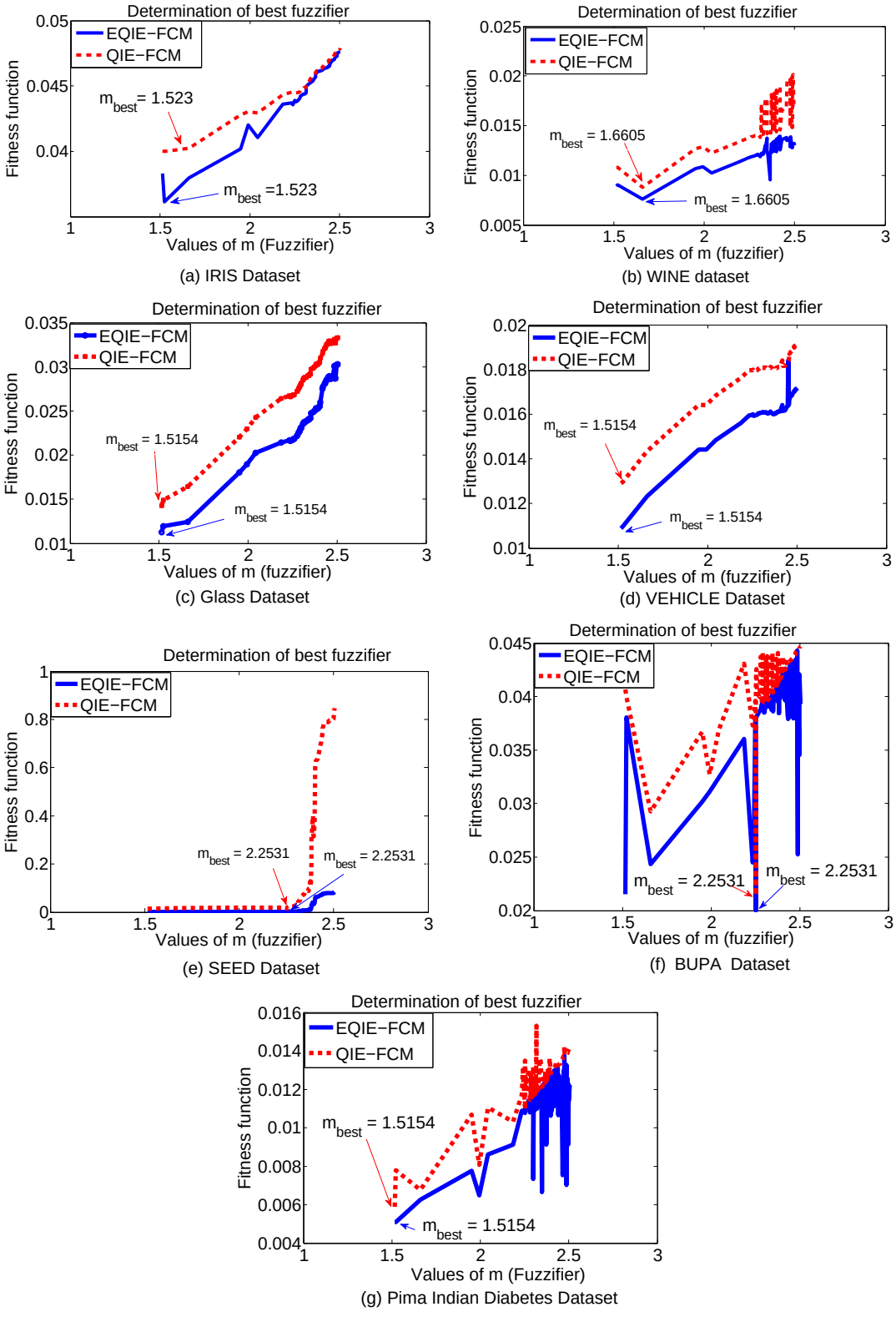


Figure 4.4: Comparison of the EQIE-FCM algorithm with QIE-FCM algorithm indicating the best value of fuzzifier for different datasets.

comparison 1.106 times lesser than the value of fitness function achieved by the QIE-FCM algorithm at $m_{best} = 1.523$. For WINE dataset, the minimum value of fitness function achieved by the EQIE-FCM algorithm at $m_{best} = 1.6605$ is comparatively 1.157 times smaller than the fitness function value attained by

the QIE-FCM at $m_{best} = 1.6605$. Similarly, the minimum value of fitness function achieved by the EQIE-FCM on GLASS, VEHICLE and PID datasets at $m_{best} = 1.5154$ is comparatively 1.266 times, 1.183 times, and 1.158 times lesser than the fitness function value attained by the QIE-FCM at $m_{best} = 1.5154$, respectively. In addition to this, on SEED and BUPA datasets, the minimum value of fitness function achieved by the EQIE-FCM at $m_{best} = 2.2531$ is 23.58 times and 1.071 times lesser than the value of fitness function attained by the QIE-FCM on these datasets at $m_{best} = 2.2531$, respectively. Although, both the algorithms for these datasets identifies the same best value of fuzzifier but the EQIE-FCM algorithm achieved a much lesser value of fitness function in comparison with the QIE-FCM algorithm. The reason for the same is discussed next. Hence, the above-reported results justify the superiority of EQIE-FCM algorithm over the QIE-FCM algorithm in terms of fitness value.

Comparison of best location of cluster centers

Table 4.7 presents the initial cluster center locations predicted by the EQIE-FCM algorithm in comparison with the randomly initialized location of cluster centers taken by the QIE-FCM algorithm. In this table, the content highlighted in bold represents the location of cluster centers find by both the algorithms corresponding to the two-dimensional scatter plot of PID dataset shown in Figure 4.3. It is observed that the best location of cluster centers predicted by the EQIE-FCM algorithm is reasonably good because each predicted location of the centers is almost in the middle of the clusters shown in Figure 4.3. However, the cluster center locations were chosen by the QIE-FCM algorithm for each cluster is almost colliding with each other. Due to the random selection of initial cluster center locations by the QIE-FCM algorithm, the clustering results achieved by this algorithm may trap into the local optima. Conversely, in the EQIE-FCM

Table 4.7: Comparison of initial cluster center locations of the EQIE-FCM Algorithm with the QIE-FCM Algorithm.

Algorithm	cluster	NTP	PGC	DBP	TSFT	SI	BMI	DPF	YOA
EQIE-FCM	1	16.9558	169.3641	6.2682	**1.2567**	347.1789	37.5227	**1.0896**	43.28969
	2	11.8800	108.8000	10.4400	**36.7589**	136.7000	45.4500	**0.8345**	55.1400
	3	4.0778	181.8700	1.0892	**49.7660**	197.5400	44.9360	**2.2782**	32.1470
QIE-FCM	1	3.38842	121.1809	69.0090	**20.9514**	81.5108	31.6374	**0.4810**	33.3568
	2	3.9476	120.9351	68.7671	**20.1197**	75.6281	32.2555	**0.4686**	33.5969
	3	3.7266	121.1631	69.5013	**20.5423**	81.5276	32.1018	**0.4683**	32.8297

algorithm, the cluster centers are initially represented in terms of a quantum bit, and after several generations of evolution, the EQIE-FCM algorithm comes out with the best location of cluster centers. As we can see in Table 4.7, cluster center locations predicted by the EQIE-FCM algorithm is more accurate than the randomly chosen location by the QIE-FCM algorithm.

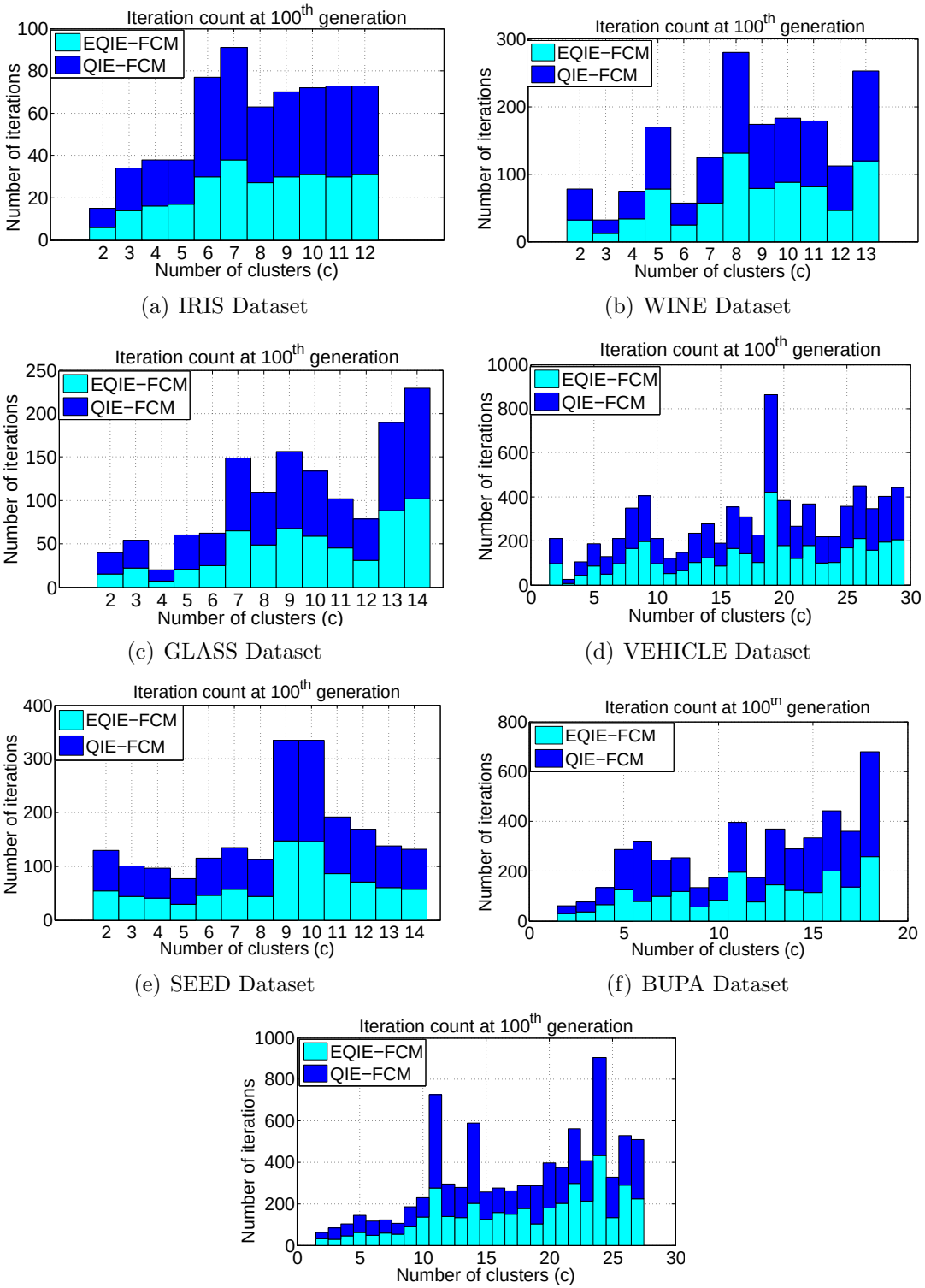
Performance comparison in terms of iterations count per cluster

The number of iterations required to find a stable location of cluster centers by the EQIE-FCM algorithm in comparison with the QIE-FCM algorithm for 100^{th} generation is reported in Figure 4.5. The results show that the proposed EQIE-FCM algorithm always takes the lesser number of iterations in comparison with the QIE-FCM algorithm for finding a stable location of cluster centers on each cluster number. The reported results show that the QIE-FCM algorithm is much more computationally intensive than the EQIE-FCM algorithm. The reason behind the better computational performance of the EQIE-FCM algorithm in comparison with the QIE-FCM algorithm is that in the QIE-FCM algorithm, the location of initial cluster centers is decided randomly. So, if the data points are located far away from the actual location of initial cluster centers, then the algorithm will converge slowly by taking many iterations to find the proper location of cluster centers. However, in the case of the EQIE-FCM algorithm, due to the selection procedure of location of initial cluster centers, it takes the less number of iterations in finding the proper location of cluster centers, and hence, results in faster convergence.

Comparison with evolutionary fuzzy clustering algorithms

To further investigate the efficacy of proposed EQIE-FCM algorithm, it is also compared with other evolutionary fuzzy based clustering algorithms [42,43,104]. Table 4.8, shows that the proposed approach is found to be significantly better than compared approaches in terms of the best value of fitness function and the optimal number of clusters. The optimal number of clusters identified by the proposed EQIE-FCM algorithm for different datasets is similar to the number of clusters can be seen according to the distribution of these datasets shown in Figure 3.4 and Figure 4.3. Moreover, the best value of fitness function achieved by the EQIE-FCM algorithm is comparatively much lesser than

the fitness value attained by the other compared approaches. Hence, the discussed results quantify the effectiveness of proposed EQIE-FCM algorithm over comparative algorithms.



(a) IRIS Dataset

(b) WINE Dataset

(c) GLASS Dataset

(d) VEHICLE Dataset

(e) SEED Dataset

(f) BUPA Dataset

(g) Pima Indian Diabetes Dataset

Figure 4.5: Performance comparison of the number of iterations taken by the EQIE-FCM algorithm and QIE-FCM algorithm.

Table 4.8: Performance comparison of the EQIE-FCM algorithm with evolutionary algorithms.

Datasets	EQIE-FCM			FCMVGA [42]			QM-FCM [43]			RQECA [104]		
	c_{best}	fitness value	SD	c_{best}	fitness value	SD	c_{best}	fitness value	SD	c_{best}	fitness value	SD
IRIS	2	0.03614	0.05291	6	4.8024	2.0938	4	0.3866	1.0283	5	0.7587	1.0922
WINE	2	0.00761	0.02883	12	4.03309	1.9739	5	0.0724	1.0982	7	0.4865	1.0083
GLASS	4	0.01127	0.01145	2	36.0576	2.0973	3	0.4201	1.9822	14	1.8975	1.0022
VEHICLE	3	0.01089	0.02463	5	48.0980	2.3473	5	1.4289	1.7362	25	4.5610	1.0237
SEED	3	0.00147	0.0098	4	8.7352	1.0983	6	2.5672	1.7382	12	1.6983	1.7282
BUPA	2	0.02000	0.06737	7	10.2568	1.0092	7	1.8956	2.3622	16	3.5689	1.0282
PID	3	0.00508	0.09821	5	0.011029	0.0382	16	0.0045	0.2919	22	0.0011	0.0181

4.4 Summary

In this chapter, mechanisms are designed to select the optimal number of clusters, the fuzzifier and the location of cluster centers. For this, we proposed two algorithms named as the QIE-FCM algorithm and the EQIE-FCM algorithm. In the QIE-FCM algorithm, an evolutionary principle of quantum computing is used to find the best value of fuzzifier in several generations from a large search space by maintaining a proper balance between exploration and exploitation. In this algorithm, the fuzzifier is evolved in each generation and the fitness is determined in each generation by using the VI_{DSO} index as the objective function. This algorithm determines the best value of fuzzifier and correspondingly it determines the optimal number of clusters after several generations of evolution. The major limitation of this approach is that it randomly initializes the location of cluster centers. Due to this, it takes a higher number of iterations to determine the actual (best) location of cluster centers, and thus it leads to a slow convergence problem. To overcome this problem, we proposed the another algorithm referred to as the EQIE-FCM algorithm that finds the optimal parameters of fuzzy clustering, i.e., the fuzzifier, the number of clusters, and the location of cluster centers from a large search space. It resolves the problem of local convergence, which occurs due to the random selection of these parameters and also overcomes the slow convergence problem of QIE-FCM.

We have compared the results of the QIE-FCM algorithm with two well-known cluster validity indexes V_{CWB} and v_{OS} in terms of the optimal value of the fitness function, the fuzzifier, and the number of clusters. The reason for comparing the proposed QIE-FCM algorithm with the V_{CWB} and v_{OS} indexes is that in the QIE-FCM algorithm, the fitness functions are formulated by using the

VI_{DSO} index that is used to evaluate the fitness of fuzzy partitions. Therefore, it is fair to compare the performance of the QIE-FCM algorithm with existing validity indexes. The results are evaluated on various datasets taken from the UCI repository. It is observed from the results that proposed QIE-FCM algorithm achieves the minimum value of fitness function, identify the best value of fuzzifier, and the optimal number of clusters for all the datasets. Furthermore, the performance of QIE-FCM is evaluated in comparison with other evolutionary algorithms. The reported results show that the QIE-FCM algorithm correctly identifies the optimal number of clusters and attain a minimum value of the fitness function in comparison with other evolutionary algorithms.

The performance of one more proposed EQIE-FCM algorithm is compared with the proposed QIE-FCM algorithm and with three other evolutionary clustering algorithms. The experimental results show that the EQIE-FCM algorithm is very effective in terms of proper selection of learning parameters, and hence, it converges to the optimal value of parameters. Due to a selection procedure of location of initial cluster centers in the EQIE-FCM algorithm as compared to the random initialization of location of cluster centers done in the QIE-FCM algorithm, the EQIE-FCM algorithm takes the lesser number of iterations in finding the proper location of cluster centers and results in a fast convergence as compared to the QIE-FCM algorithm. This verifies the effectiveness of the proposed EQIE-FCM algorithm over compared approaches. These algorithms enhance fuzzy clustering capability with proper selection of clustering parameters. Thus, leads to the sophisticated design of fuzzy clustering algorithms.

Chapter 5

Design of Incremental Clustering Algorithm as a Classifier for Handling Very Large Data

5.1 Introduction

Nowadays, with the advancement in technology, an enormous amount of data is continuously generated every day from various sources, including e-commerce, social networks, financial records, medical records, population databases; or from scientists who routinely take large-scale simulations for weather prediction or drug discovery. This kind of progressively generated data is considered as valuable resources since it can provide key insights in many areas such as human behavior analysis, market trends, disease identification, engineering safety, and environmental change [163, 164]. Mining valuable information in prevalent VL data is vital to gain key insights in voluminous data to get an edge in the competitive market.

Clustering is a promising data-analysis technique that aims to organize a collection of patterns into groups for finding pattern structure and information underlining unlabeled data. Clustering has been successfully applied to the analysis of datasets in various fields such as gene analysis [14], community detection [15], image processing [35], and analysis of microarray data in bioinformatics [165]. For clustering of VL data, one of the biggest challenges is that data is too large to be stored in memory. In this case, clustering al-

gorithms that need to store all the data in memory become infeasible. Like, the literal fuzzy clustering algorithm [22, 36] as discussed in section 2.1.2, performs clustering of the entire data. Therefore, clustering of VL data with this algorithm becomes infeasible due to space and time bottleneck. To handle VL data extended clustering schemes [29, 36] have been proposed in the past, as discussed in section 2.5. The extended clustering algorithms applied clustering to a representative (and manageably sized) sample of full dataset, and then it noniteratively extends sample results to obtain clusters for remaining data of the entire dataset. As discussed in section 2.5, the rseFCM algorithm works on a representative sample of full dataset and then the locations of cluster centers, produced on sampled data, have been extended to compute partitions with full data. The rseFCM algorithm suffers from overlapping locations of cluster centers, which significantly affects clustering results. The problem of overlapping location of cluster centers arises because sampled data does not adequately cover all samples present in VL data, and thus, it is unable to appropriately capture the entire sample space represented by the VL data. Hence, the error between the locations of cluster centers produced by the sampled data and the locations generated by clustering the entire VL data become drastically higher. Thus, the rseFCM algorithm approximately minimizes the objective function, which drastically deviates from the minimum value of objective function produced by clustering VL data at once.

Recently, incremental clustering framework [36] has been adapted to perform clustering of VL data by processing data chunk by chunk, whereas one chunk represents a small portion of the entire data. However, the chunk size can be decided by the user. Therefore, to address clustering of VL data and to overcome the drawback of the rseFCM algorithm, in this chapter, we propose an incremental clustering algorithm named as RSIO-FCM, which is discussed in detail in the further section. Next, the Bayes' theorem [153] is employed to design a classification mechanism, which relies on the clustering results generated by the RSIO-FCM algorithm. The designed classification mechanism is dependent on the cluster centers, and it assigns labels to the formed clusters that are used further to predict the class labels of unknown samples. The details of the designed classification mechanism are presented in the further section, but before this, we present the preliminaries of Bayes' theorem.

5.2 Preliminaries

As discussed earlier in section 2.1.2, fuzzy clustering algorithm works on the collection of n data points x_i, $i = 1, ..., n$ and it partitions these data points into c fuzzy clusters. Thus, it gives the outcomes as cluster centers for c fuzzy clusters and membership degree corresponding to the n data points. Then, a classifier is designed using Bayes' theorem [153], which is termed as a Bayesian classifier. This is a statistical classifier that can predict class membership probabilities with which a given data point belongs to the class. According to the probability theory and statistics, Bayes' theorem describes the probability of a data point by considering both the prior probability and conditional probability of a data point to determine the posterior probability of a data point.

Bayes' theorem is stated mathematically as follows:

$$P(H|x_i) = \frac{P(x_i|H)P(H)}{P(x_i)} \tag{5.1}$$

where x_i is the i^{th} data point and H is some hypothesis.

- $P(H)$ is the prior probability of H, the probability that H is correct before that data x_i is seen.
- $P(x_i|H)$ is the conditional probability of observing the data point x_i given that the hypothesis H is true.
- $P(x_i)$ is the marginal probability of x_i.
- $P(H|x_i)$ is the posterior probability, the probability that the hypothesis is true, given the data and the previous state of belief about the hypothesis.

In general, the definitions of prior probability, conditional probability, marginal probability, and posterior probability are defined as follows:

- **Prior probability -** The prior probability of a data point is the likelihood of a data point computed before the collection of new data. For example, if 0.01 percent of a population has Alzheimer's disease than the probability that a person has drawn at random would have Alzheimer's is 0.01.
- **Posterior probability -** The posterior probability of a data point is the likelihood of a data point computed following the collection of new data.

One begins with a prior probability of a data point and revises it in the light of new data, so the adjusted probability is the Posterior probability.

- **Conditional probability -** It is the probability of the presence of a data point x_i given that hypothesis H is true is called the conditional probability of x_i given hypothesis H. For example, the probability it rains on Tuesday given that it rained on Monday would be written as $P(Rain\ on\ Tuesday|Rain\ on\ Monday)$.

- **Marginal probability -** It is the probability of the presence of a data point without depending on other data points, and it may be thought of as an unconditional probability.

- **Joint probability -** The joint probability is a statistical measure of the presence of two data points at the same time, i.e., a data point x_i occurring at the same time data point x_{i+1} occurs.

- **Total probability -** According to probability theory, the law of total probability is a fundamental rule relating marginal probabilities to conditional probabilities. If x_1, x_2, $x_3...x_n$ is a partition of sample space, then for any hypothesis H we have

$$P(H) = \sum_{i=1}^{n} P(H|x_i)P(x_i) \tag{5.2}$$

The Bayes' theorem presented in Eq. (5.1) and the total probability concept given in Eq. (5.2) is used for designing a classifier when clustering of input data points is over. Thus, the probabilities given in Eq. (5.1) is evaluated by the presence of data points in a cluster and as well as belongingness of data points to a particular class. The Eq. (5.1) establishes a relationship between clusters and classes. The details are presented in the next section.

5.3 Proposed Work

This section presents the detailed description of the proposed RSIO-FCM algorithm to perform clustering of VL data. Furthermore, we introduce the design

of an efficient classifier to label the formed clusters with the RSIO-FCM algorithm. The classifier is designed by using the concept of Bayes' theorem [153], which utilizes the outcomes of RSIO-FCM clustering algorithm to perform the classification of VL data.

5.3.1 Design of Random Sampling Iterative Optimization Fuzzy C-Means Algorithm

The proposed RSIO-FCM is an incremental clustering algorithm, which performs clustering of VL data by processing data chunk by chunk and it is designed to overcome the drawbacks of an extended clustering algorithm called as rseFCM [36]. The proposed algorithm processes the VL data by dividing it uniformly into various chunks or subsets such as $X = \{X_1, X_2, \ldots\ldots X_s\}$, where s denotes the number of subsets. These subsets are created such that each subset contains distinct data points. It means that the data points present in one subset will not be included in the rest of the subsets. If the data points present in all the subsets are combined, then we get the complete VL data. Also, these subsets jointly represent the same sample space that is represented by a VL data. Therefore, we can assure that all the subsets adequately cover all the data points present in the VL data. In this approach, cluster centers for clustering of the first subset are chosen randomly from the entire data. The RSIO-FCM algorithm starts by performing clustering of the first subset. Then, final cluster centers and membership matrix for the first subset denoted as V_1 and U_1, respectively, are found by applying the LFCM algorithm [36]. After clustering of the first subset, cluster centers of the first subset V_1 are used as the initial cluster centers for clustering of the second subset. The same process is repeated for the clustering of all the subsets. The final cluster centers for entire (VL) data is determined by combining the membership matrix of all the subsets and using Eq. (2.8). Thus, the proposed RSIO-FCM algorithm generates non-overlapping locations of cluster centers and works significantly well by covering the entire sample space represented by VL data. The outline of the algorithm is presented next in the form of ***Algorithm 5.1***.

Algorithm 5.1 Algorithm for Random Sampling Iterative Optimization Fuzzy C-Means to Iteratively Minimize $J_m(U, V^{'})$

1: **Input** : X, V, c, m, ϵ; X represents a dataset consists of n data points such that $X = \{x_1, x_2, ...x_n\}$, V is the set of initial cluster centers represented as $V = \{v_1, v_2, ...v_c\}$, c denotes the number of clusters, m represents fuzzifier, and ϵ represents the termination criteria.
2: **Output**: $U, V^{'}$; $V^{'}$ denote the set of final cluster centers and U represents the membership matrix.
3: **Begin**
4: **Load** X as n_s sized randomly chosen subsets $X = \{X_1, X_2 \ldots\ldots X_s\}$.
5: **Sample** X_1 from X without replacement.
6: $U_1, V_1 = LFCM(X_1, V, c, m)$
7: **for** $e = 2$ to s do
8: $\quad U_e, V_e = LFCM(X_e, V_{e-1}, c, m)$
9: **end for**
10: **Compute** the partition on full dataset.

$$U = \sum_{e=1}^{s} U_e \tag{5.3}$$

11: **Compute** cluster centers $V^{'}$ using Eq. (2.8) in which U is utilized from Eq. (5.3).
12: **Compute** the objective function using Eq. (2.6).
13: **Return** $U, V^{'}$
14: **End**

The RSIO-FCM algorithm adequately covers the sample space and generates final locations of cluster centers by feeding forward the cluster centers of one subset for processing of the next subset. Thus, the final locations of cluster centers produced by the RSIO-FCM algorithm is as if the clustering performed on the entire dataset at once. Due to the proper locations of cluster centers generated by the RSIO-FCM algorithm, it minimizes the objective function in a better way as compared to the rseFCM algorithm. This is because the rseFCM algorithm performs clustering only on a subset of the entire data and extends the clustering results to produce the clustering results of the entire dataset. Thus, the final clustering results obtained by the rseFCM do not cover the entire sample space. Therefore, it approximately minimizes the objective function.

To show the effectiveness of clustering results generated by the RSIO-FCM algorithm, it is further designed as a classifier by utilizing the concept of Bayes' theorem [153] so that the clusters produced by the RSIO-FCM algorithm are assigned class labels. Since classifier is designed with Bayes' theorem which is probability-based, therefore the designed classification system is tested based

on probability. Thus, the classification results are dependent on the clusters generated by the RSIO-FCM algorithm. Therefore clustering results have a significant impact on classification results. The detailed description of the same is presented next.

5.3.2 Design of Classifier for RSIO-FCM Clusters

After clustering of VL data with the RSIO-FCM algorithm, it is designed as a classifier by utilizing the concept of Bayes' theorem [153], which assigns class labels to the formed clusters generated through the RSIO-FCM algorithm and then the labeled clusters are used to predict the class labels of unknown test samples. The designed classifier is dependent only on the cluster centers emphasizes that the locations of cluster centers have a significant impact on classification results. To assign class labels to the formed clusters a relational matrix ($\mathcal{P}$) is modeled by using Bayes' theorem, which establishes cluster class relationships. The $\mathcal{P}$ constitutes as a $c \times w$ matrix, which is defined as follows:

$$\mathcal{P} = \begin{bmatrix} P(class_1|cluster_1) & P(class_2|cluster_1) & \cdots & P(class_w|cluster_1) \\ P(class_1|cluster_2) & P(class_2|cluster_2) & \cdots & P(class_w|cluster_2) \\ \cdots & \cdots & \cdots & \cdots \\ P(class_1|cluster_c) & P(class_2|cluster_c) & \cdots & P(class_w|cluster_c) \end{bmatrix} \tag{5.4}$$

Each component of the $\mathcal{P}$, i.e., $P(class_l/cluster_j)$ shows that the probability of l^{th} class having its belongingness to j^{th} cluster is computed as follows:

$$P(class_l|cluster_j) = \frac{P(class_l, cluster_j)}{P(cluster_j)} \tag{5.5}$$

where $cluster_j$ represents the j^{th} cluster, $j = 1, 2, .., c$, and $class_l$ denote the l^{th} class, and $l = 1, 2, .., w$, $P(cluster_j)$ represents the prior probability and it can be rewritten as $Num(x_i \in cluster_j)/n$, $\forall i \in [1, n]$, which determines the number of samples (Num) belonging to the j^{th} cluster ; $P(class_l, cluster_j)$ represents the joint distribution, and it can be rewritten as $Num(x_i \in class_l$ and $x_i \in cluster_j)/n$, $\forall i \in [1, n]$, which is computed in terms of the proportion of number of samples belonging to both the j^{th} cluster and the l^{th} class. Thus, Eq. (5.5) can be rewritten and defined as follows:

$$P(class_l|cluster_j) = \frac{Num(x_i \in class_l, x_i \in cluster_j)}{Num(x_i \in cluster_j)}; \forall i \in [1, n] \tag{5.6}$$

The denominator of Eq. (5.6), i.e., $Num(x_i \in cluster_j)$ is described by membership degree (u_{ij}), $i = 1, .., n$, and $j = 1, ..., c$, and is computed using Eq. (2.7). Thus, for all the clusters $j = 1, .., c$ the u_{ij} or $Num(x_i \in cluster_j)$ is computed using Eq. (2.7) to get a U of size $n \times c$. The Eq. (2.7) shows that clustering results are dependent on the cluster centers. Also, the assignment of class labels to the formed clusters in Eq. (5.6) is forming a cluster class relational matrix using Eq. (5.4) which also shows the dependency on cluster centers. Thus, the cluster class relational matrix determines the statistical relationship between the formed clusters and the given classes, and it is used subsequently to assign labels to test samples. Therefore, locations of cluster centers have a significant impact on classification results, which is discussed next.

5.3.3 Testing of Designed Classifier

Once the classifier is designed for a given set of samples, testing is carried out using the total probability theorem, and a cluster class relational matrix is computed using Eq. (5.4). The details are as follows.

Given a test sample x_i, the belongingness of the sample x_i to c clusters is determined by u_{ij} or $Num(x_i \in cluster_j)$, $\forall j \in [1, c]$, and is computed using Eq. (2.7). Then, the output class labels of the test sample x_i are represented by $f(x_i)$ and are computed by using $\mathcal{P}$ and u_{ij} evaluated using Eqs. (2.7), (5.4), and (5.5), respectively. The output class labels of the test sample x_i are determined through the total probability theorem [153], which is defined as:

$$P(class_l|x_i) = \sum_{j=1}^{c} P(cluster_j|x_i)P(class_l|cluster_j) \tag{5.7}$$

where $P(cluster_j|x_i)$ represents the posterior probability of the presence of corresponding samples in the j^{th} cluster and can be computed using Eq. (2.7), and $P(class_l|cluster_j)$ denotes the posterior cluster probabilities of class membership, i.e., the presence of samples that belong to both the l^{th} class and j^{th} cluster and can be computed using Eq. (5.5). Thus, the output class labels $f(x_i)$ is defined as

$$f(x_i) = \underset{1 \le l \le L}{\operatorname{argmax}} P(class_l|x_i) \tag{5.8}$$

It can be seen from Eqs. (5.7) and (5.8), the output class labels of the test

sample x_i, i.e., $f(x_i)$ is also showing its dependency on clustering results, i.e., cluster centers. Thus, it can be inferred from this, that clustering results are significantly affecting the classification results. The complete clustering and classification process is presented diagrammatically in Figure 5.1.

An illustrative example is used here to describe the steps involved in the clustering and classification procedure. For this, we have taken IRIS data from UCI Machine Learning Repository [154], and it is discussed in Appendix A. Half of the data are used for training and the remaining half is used for testing purposes. The training data are randomly divided into three subsets or chunks, and the cluster centers for them are randomly initialized. The locations of these cluster centers are presented in Table 5.1. Then, with these cluster centers, clustering is performed on the training data using the RSIO-FCM algorithm, as stated in Algorithm 5.1. As soon as clustering of training data are over, the final membership matrix, i.e., the denominator of Eq. (5.5), i.e., $P(cluster_j)$ is obtained that shows the belongingness of each sample to a particular cluster.

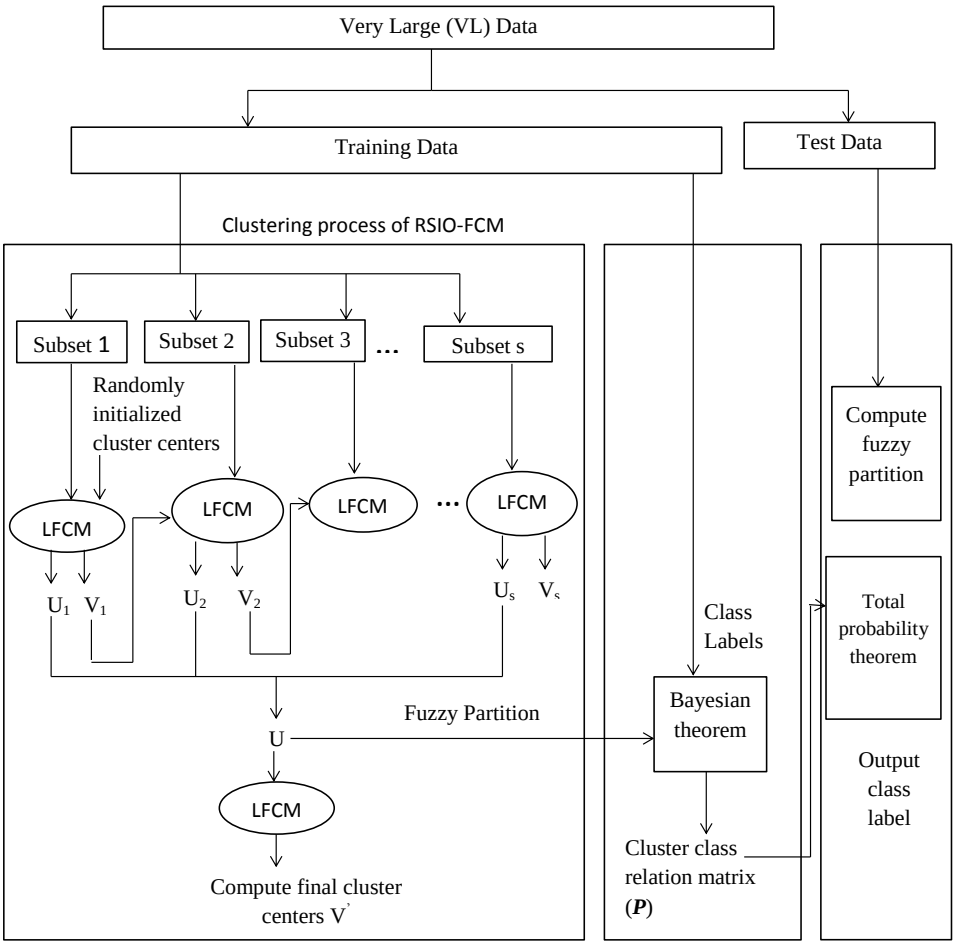


Figure 5.1: Flowchart of the RSIO-FCM clustering and its classification process.

Table 5.1: Information of IRIS data with three classes in three groups.

Group	class label	cluster centers
1	1	(5.0155, 3.414, 1.5576, 0.24494)
2	2	(6.3242, 2.8909, 5.295, 1.9086)
3	3	(5.7586, 2.6568, 4.1103, 1.2663)

Each sample shows its belongingness to a particular cluster, based on higher membership degree as depicted in Table 5.2. The final locations of cluster centers are obtained after clustering of training samples of IRIS data with the RSIO-FCM algorithm, as reported in Table 5.3 and visualized in Figure 5.2.

The class membership degree corresponding to the training data is already known in advance and given in Table 5.4. Thus, we can infer from Tables 5.2

Table 5.2: Membership matrix of training samples of IRIS data.

Sample	Cluster		
	1	2	3
1	**0.69779**	0.13879	0.16342
2	**0.67990**	0.14767	0.17243
3	**0.60925**	0.18106	0.20970
4	**0.60060**	0.18374	0.21566
5	**0.59959**	0.1828	0.21761
.	.	.	.
.	.	.	.
26	0.20312	**0.46057**	0.33631
27	0.14875	**0.55831**	0.29294
28	0.19461	0.25635	**0.54904**
29	0.16344	**0.51696**	0.3196
30	0.19733	**0.50066**	0.30201
.	.	.	.
.	.	.	.
71	0.15997	0.26076	**0.57927**
72	0.17836	0.35608	**0.46556**
73	0.19721	0.29326	**0.50953**
74	**0.54307**	0.20835	0.24857
75	0.23958	0.26591	**0.49451**

Table 5.3: Final locations of cluster centers of IRIS Data.

Group	Class label	Final locations of cluster centers
1	1	(5.6644, 3.1459, 3.3512, 1.0843)
2	2	(6.0146, 3.0834, 3.9074, 1.2993)
3	3	(5.957, 3.0913, 3.9417, 1.2933)

and 5.4, that sample 1 belongs to cluster 1 and class 1. Likewise, to determine the numerator of Eq. (5.5), we have to count the total number of samples, which belongs to both cluster 1 and class 1. Thus, a similar observation is carried out for other pairs of clusters and classes, respectively. Finally, the cluster class

Table 5.4: Information of class membership matrix of training samples of IRIS data.

Sample	Classes		
	1	2	3
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	1	0	0
.	.	.	.
.	.	.	.
26	0	1	0
27	0	1	0
28	0	1	0
29	0	1	0
30	0	1	0
.	.	.	.
.	.	.	.
71	0	0	1
72	0	0	1
73	0	0	1
74	0	0	1
75	0	0	1

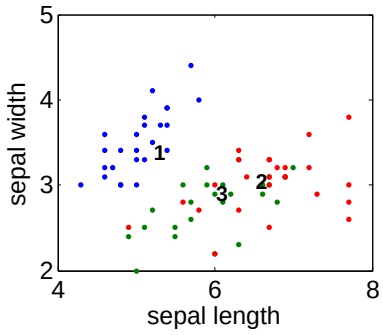


Figure 5.2: Graphical representation of final locations of cluster centers of IRIS data.

relational matrix of training data is obtained as follows:

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.095238 & 0.90476 \\ 0 & 0.7931 & 0.2069 \end{bmatrix} \tag{5.9}$$

Now, the relational matrix is used to determine the class labels of test samples. For this, first the cluster membership degree of each test sample, i.e., $P(cluster_j|x_i)$ given in Eq. (5.7) is computed for all values of $j = 1, ..., c$ using Eq. (2.7) and the obtained values are reported in Table 5.5. Now, the obtained cluster class relational matrix along with the cluster membership matrix of test samples given in Table 5.5 is utilized in Eq. (5.7) for classification of test samples. Thus, the output class label of test samples is determined and presented in Table 5.6. Next, the output class label of test samples reported in Table 5.6 is defuzzified using Eq. (5.8) and thus, the defuzzified output class label of test samples is reported in Table 5.7.

Table 5.5: Membership matrix of test samples of IRIS data.

Sample	Clusters		
	1	2	3
1	**0.36764**	0.31645	0.31591
2	**0.36732**	0.31653	0.31616
3	**0.36799**	0.31615	0.31586
4	**0.36823**	0.31617	0.3156
5	**0.36908**	0.31613	0.3148
.	.	.	.
.	.	.	.
26	0.25673	**0.37248**	0.37079
27	0.26728	**0.36759**	0.36513
28	0.26679	**0.37016**	0.36306
29	0.28487	**0.35763**	0.35749
30	0.24615	**0.38078**	0.37307
.	.	.	.
.	.	.	.
71	0.29352	0.35108	**0.35540**
72	0.28863	0.35530	**0.35608**
73	0.28173	0.35794	**0.36033**
74	0.29583	0.35056	**0.35361**
75	0.28503	0.35491	**0.36005**

Table 5.6: Output class labels of test samples of IRIS data.

Sample	Classes		
	1	2	3
1	**0.36764**	0.28069	0.35168
2	**0.36732**	0.28089	0.3518
3	**0.36466**	0.28226	0.35308
4	**0.36799**	0.28062	0.35139
5	**0.36823**	0.28042	0.35136
.	.	.	.
.	.	.	.
26	0.25673	**0.41372**	0.32955
27	0.31218	**0.38038**	0.30744
28	0.26728	0.32459	**0.40813**
29	0.26749	0.33035	**0.40216**
30	0.26679	**0.41002**	0.32319
.	.	.	.
.	.	.	.
71	0.28173	0.31987	**0.39840**
72	0.2751	0.32276	**0.40214**
73	0.31055	0.30608	**0.38337**
74	0.28968	0.31713	**0.39319**
75	0.28503	0.31936	**0.39561**

Table 5.7: Defuzzified output class labels of test samples of IRIS data.

Sample	Classes		
	1	2	3
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	1	0	0
.	.	.	.
.	.	.	.
26	0	1	0
27	0	1	0
28	0	0	1
29	0	0	1
30	0	1	0
.	.	.	.
.	.	.	.
71	0	0	1
72	0	0	1
73	0	0	1
74	0	0	1
75	0	0	1

5.4 Experimental Evaluation

In this section, we perform the experiments to investigate the clustering and classification capability of the proposed RSIO-FCM algorithm on four benchmark VL datasets. The performance of proposed RSIO-FCM is evaluated in comparison to the existing rseFCM algorithm [36] in terms of classification accuracy, run-time, and objective function. The information about datasets and experimental settings are presented next.

5.4.1 Datasets and Experimental Settings

To demonstrate our approach, we have taken four benchmark VL datasets namely Pen-Based Recognition of Handwritten Digits, Page Blocks Classification, HRTU2, EEG Eye State from UCI Machine Learning repository [154]. The detailed description of these datasets is given in Appendix A. To evaluate the performance of proposed RSIO-FCM algorithm in comparison with the rseFCM algorithm, we have divided the training and testing data into 50-50 training-testing partition. It means that every dataset is partitioned into two halves, where one-half of the dataset is used for training, and the other half of the dataset is used for testing purposes. All the experiments are carried out on Intel(R) Xeon(R) E5-1607 Workstation PC with 64 GB of memory and running on the Windows 7 professional operating system with a processing speed of 3.10 GHz. Both the approaches are implemented in MATLAB computing environment and executed on MATLAB version R2014a. In the case of both the algorithms, we have taken the number of clusters equal to the number of classes of the dataset according to the study suggested by the researchers [36, 45, 166]. Thus, the value of c taken for each dataset is reported in Table 5.8. For each dataset, we have randomly chosen c data points from a dataset as the initial

Table 5.8: Information of number of clusters taken for VL datasets

Datasets	c
Pen-Based Recognition of Handwritten Digits	10
Page Blocks Classification	5
HRTU2	2
EEG Eye State	2

cluster centers to initialize V. In all of our experiments, we fix the value of $\epsilon = 10^{-3}$ [167].

5.4.2 Experimental Results and Discussion

In this section, we present experimental results to test the performance of the proposed RSIO-FCM on a different variety of VL datasets, as well as compare its performance with the rseFCM approach. The performance of RSIO-FCM in comparison with rseFCM is measured in VL data in terms of various parameters; (i) the fuzzifier, which controls the degree of fuzziness among fuzzy clusters and drastically affects cluster formation, (ii) the objective of both the algorithms is to minimize the objective function iteratively, (iii) run time (reported in seconds) is the time required to group the VL data into clusters and to compute cluster centers, and (iv) the classification accuracy determines the number of correctly classified samples.

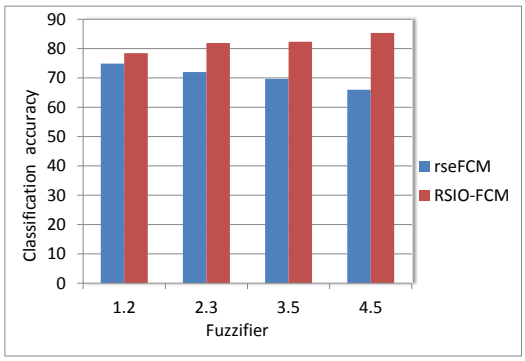
From the experimental results reported in Table 5.9, it is observed that the RSIO-FCM algorithm minimizes the objective function for all the datasets in a better way as compared to the rseFCM algorithm. Also, the rseFCM algorithm generates improper locations of cluster centers. Therefore, it results in degradation of classification accuracy. In contrast to rseFCM, the RSIO-FCM algorithm generates proper locations of cluster centers for every subset of a dataset because it covers the entire sample space during clustering. Thus, the RSIO-FCM algorithm produces proper locations of cluster centers due to which classification accuracy enhances with the increase of fuzzifier (m) by continuously minimizing the objective function. In addition to this, it requires lesser run-time in comparison with the rseFCM algorithm. Thus, the RSIO-FCM is preferable for handling VL data.

As shown in Figure 5.3, the classification accuracy of the RSIO-FCM and the rseFCM is evaluated on four different values of m. It can be observed that with an increase in the value of m, the classification accuracy achieved by the RSIO-FCM algorithm continuously increases and much higher than the accuracy obtained by the rseFCM algorithm on all the VL datasets. Thus, it can be inferred from the reported results that higher value of m is suitable for classification of VL data. The same is verified by the results reported in Table 5.9

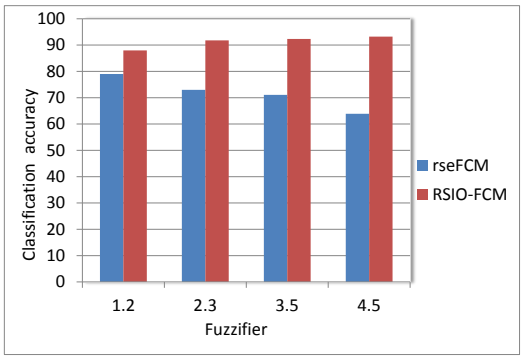
that the RSIO-FCM algorithm shows drastic improvement in classification performance of VL data as compared to the rseFCM algorithm on four benchmark VL datasets.

Table 5.9: Comparison of the rseFCM and the RSIO-FCM Algorithms on benchmark VL datasets.

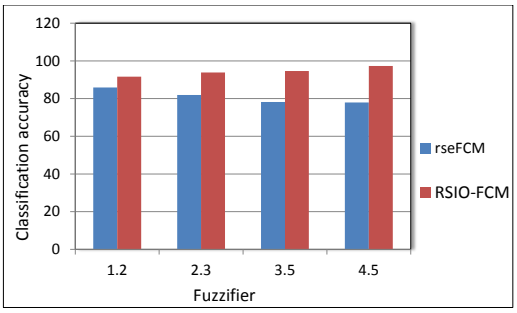
Datasets	Algorithms	Parameters			
		m	Objective Function	Run-time (seconds)	Accuracy (%)
Pen-Based Recognition of Handwritten Digits	rseFCM	1.2	7833685633	31.79	74.88
		2.3	11917813.07	3.40	71.99
		3.5	13718180.49	3.38	69.67
		4.5	281935.08	2.42	65.96
	RSIO-FCM	1.2	4622236.18	28.56	78.45
		2.3	10917812.07	2.089	81.89
		3.5	1271610.49	1.892	82.33
		4.5	211935.08	1.320	85.34
Page Blocks Classification	rseFCM	1.2	387910619.6	1.54	79.02
		2.3	181503981.8	1.44	72.98
		3.5	79004007.83	1.43	71.09
		4.5	39501984.9	1.36	63.9
	RSIO-FCM	1.2	58137603.98	0.560	87.96
		2.3	12096408.21	0.495	91.77
		3.5	772374.015	0.484	92.34
		4.5	70788.88	0.480	93.22
HTRU2	rseFCM	1.2	71022001.51	54.58	85.87
		2.3	40348631.22	48.89	81.90
		3.5	920548.22	42.22	78.22
		4.5	301881.79	27.81	77.91
	RSIO-FCM	1.2	55085301.11	50.58	91.64
		2.3	11745674.78	45.34	93.87
		3.5	860761.41	38.81	94.66
		4.5	112337.20	24.32	97.31
EEG Eye State	rseFCM	1.2	154867234555.13	30.88	80.66
		2.3	25472901122.91	25.67	79.21
		3.5	234564123.21	22.45	77.89
		4.5	211381108.09	18.56	75.30
	RSIO-FCM	1.2	120879423521.92	27.66	82.98
		2.3	12273075178.23	20.78	86.71
		3.5	1012151236.69	18.77	89.91
		4.5	126518904.58	12.34	93.45



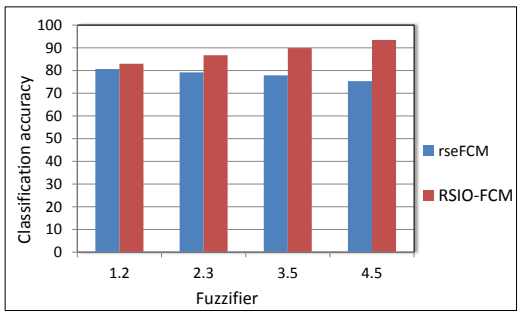
(a) Pen-Based Recognition of Handwritten Digits.



(b) Page Blocks Classification.



(c) HTRU2.



(d) EEG Eye State.

Figure 5.3: Accuracy comparison of the rseFCM and the RSIO-FCM Algorithms on benchmark VL datasets.

5.5 Summary

In this chapter, looking towards the current need of handling VL data, we proposed an incremental clustering algorithm called RSIO-FCM, which processes data, chunk by chunk. The proposed approach is designed to overcome drawbacks of the existing rseFCM algorithm, which generates overlapping locations of cluster centers. In the proposed approach, we perform clustering of the entire dataset in the following manner. First, the RSIO-FCM algorithm divides VL data into various chunks or subsets. Then, it processes data, chunk by chunk and performs clustering on each chunk by feeding forward final cluster centers of that chunk for clustering of the next chunk. The same procedure is repeated for clustering of all the chunks. Thus, RSIO-FCM algorithm covers the entire sample space adequately and generates nearly similar locations of cluster centers as produced by clustering the entire data. Furthermore, a classification mechanism is designed by employing the Bayes' theorem, which assigns class labels to the formed clusters obtained using the RSIO-FCM which is used further to determine the class label. The classification mechanism designed using the Bayes'

theorem is dependent on cluster centers. Thus the locations of cluster centers have a significant impact on classification results.

The proposed RSIO-FCM overcomes the drawbacks of the rseFCM algorithm, which fails to cover the entire sample space adequately. It is also observed that, due to proper locations of cluster centers generated by the RSIO-FCM algorithm, it achieves better classification accuracy as compared to the rseFCM algorithm. The same is verified with the experimental evaluation. The performance of proposed RSIO-FCM is evaluated on four benchmark VL datasets taken from UCI Machine Learning Repository, and its performance is evaluated in comparison with the rseFCM, which shows that RSIO-FCM achieved significant improvement in classification accuracy with a reduction in run-time. Thus, it is inferred that the accuracy of classifier significantly depends on clustering efficiency. One of the major limitation of the RSIO-FCM algorithm is that it produces highly deviated cluster centers if two chunks would consist of samples belonging to distinct classes. Therefore, the cluster centers of one chunk may be significantly different from the cluster centers of the other chunk. Due to this reason, if the cluster centers of the previous chunk are fed as an input for the clustering of the current chunk, then the number of iterations during clustering of current chunk increase significantly and leads to a slow convergence. This issue is tackled further by proposing an advanced approach in the next chapter.

Chapter 6

Scalable Clustering Algorithms for Handling Big Data

6.1 Introduction

An enormous amount of data accumulated at an accelerating rate from various areas including health care, education, e-commerce, social networking, finance, and education, due to the involvement of humans in the digital space. For example, Facebook stores more than 30 Petabytes of data, and Walmarts databases contain more than 2.5 Petabytes of data [164]. Such a huge data containing useful information is called Big Data. Thus, data nowadays is not limited to VL data, but it is extended to Big Data. It is becoming necessary to mine such Big Data to gain insights the valuable information that can be of great use in scientific and business applications. Clustering is a promising data mining technique that is widely adopted for mining valuable information from underlying Big Data. So far, many parallel clustering algorithms have been proposed by researchers [29,34,36,45,135]. All these clustering algorithms have the following drawbacks: a) They assume that all samples can reside in main memory at the same time; b) The parallel systems have provided restricted programming models and used the restrictions to parallelize the computation automatically. It is often impossible for a single processor computer to store the whole dataset in the main memory for processing, and the extensive amount of disk becomes a bottleneck in efficiency. Despite the rapid development of the clustering algorithms aimed at handling large data, there is a lack of adoption of these techniques in

the wider data mining and other application communities for Big Data problems. A likely reason for this is that these algorithms are not scalable to handle Big Data. Thus, the surging volume of Big Data and the increasing computational requirement have put tremendous pressure on clustering algorithms to scale beyond a single machine, due to both space and time bottlenecks. Therefore, scalability of the clustering algorithm is becoming an important issue [41]. To design such scalable algorithms, Big Data analytics frameworks are required. Recently, a large number of computing frameworks [120–123, 128, 129, 137, 138] have been developed for Big Data analysis. With the recent development of Big Data computing frameworks, there is an immense need to scale clustering algorithms on Big Data computing framework to achieve high performance without affecting the quality of clustering. For this reason, we present the design of scalable fuzzy clustering algorithms implemented in the Apache Spark framework for handling Big Data.

6.2 Proposed Scalable Fuzzy Clustering Algorithms for Big Data

In this section, we are presenting a proposed novel SRSIO-FCM algorithm. It is designed to deal with the challenges associated with fuzzy clustering for handling Big Data and to overcome drawbacks of the RSIO-FCM algorithm presented in chapter 5. Similar to RSIO-FCM, the proposed approach starts by randomly partitioning the data into various subsets. In this method, for clustering of the first subset, cluster centers are initialized randomly. After clustering of the first subset, the cluster centers, and membership information corresponding to the first subset is obtained. The final cluster centers obtained after clustering of the first subset is used as an input for clustering of the second subset. After clustering of the second subset, the cluster centers, and membership information is obtained. But unlike RSIO-FCM, it does not use these cluster centers as an input for clustering of the third subset. Instead, it combines the membership information of first and second subset to compute the new cluster centers. These cluster centers are then fed as an input for clustering of the third subset. Then, after clustering of this subset, the cluster centers, and membership information

corresponding to it is found. The same procedure is repeated for the clustering of the rest of the subsets. In comparison with RSIO-FCM, the proposed SRSIO-FCM uses the cluster centers obtained with the combined membership information of all the processed subsets for clustering of the current subset. This is because, in RSIO-FCM, where the cluster center of one subset is fed as input for clustering of the next subset, it might be possible that two subsets would consist of samples belong to a distinct class. Due to this reason, the cluster center of one subset fed as an input for clustering of the current subset will result in slow convergence by taking a higher number of iterations for that subset. Furthermore, it also produces the highly deviated cluster centers for that subset. This problem has been overcome in the proposed SRISO-FCM because this approach uses the combined membership information of all the processed subset which covers a wider sample space. Thus, using the cluster centers obtained with the membership information of wider sample space will avoid the problem of using the highly deviated cluster for clustering of the current subset. In addition to this, it also results in faster convergence by taking less number of iterations for clustering of the current subset. The proposed SRSIO-FCM algorithm is implemented in the Apache Spark framework, and its detailed description is presented in section 6.2.2. For the sake of performance comparison of this proposed algorithm, we designed and implemented the scalable version of existing clustering algorithms, i.e., rseFCM and LFCM on the Apache Spark cluster. The scalable version of the rseFCM and LFCM algorithm is termed as SrseFCM and SLFCM. Before presenting the details of the proposed SRSIO-FCM algorithm, we present the working of the scalable model of LFCM algorithm. The details of the SLFCM algorithm are presented next.

6.2.1 Enhanced Scalable Version of LFCM Algorithm to Handle Big Data

To design the scalable version of the LFCM algorithm implemented on the Apache Spark cluster, we first need to identify the computations in the LFCM algorithm, which could be executed in a parallel manner and the computations which could be executed only in a serial manner. As discussed earlier in section 2.6.2, chapter 2, the spark cluster consists of a master node and $w^{'}$ worker

nodes. Thus, the computations of LFCM algorithms that could be executed in a parallel manner are processed on $w^{'}$ worker nodes and the computation executed in a serial manner are processed on a master node. To identify the parallel and serial computations, first, the base algorithm, i.e., LFCM (***Algorithm 2.2***) presented in section 2.1.2 is analyzed. As presented in ***Algorithm 2.2***, the most computation intensive part of LFCM is the calculation of membership degrees of each point corresponding to all the cluster centers. For the computation of membership degree, only the data point and cluster centers values are required. As observed in Line 5 of ***Algorithm 2.2***, computation of the membership degree of each data point is independent of other data points. Therefore, computation of the membership degree of different data points can be performed in parallel on $w^{'}$ worker nodes. To perform the parallel computation of the membership degree of all the data points on $w^{'}$ worker nodes, in Line 5 of ***Algorithm 6.1***, we present the use of map function, and then the final membership degree is evaluated using a reduceByKey function.

As discussed earlier in section 2.6.2, chapter 2, the Apache Spark cluster uses HDFS [134] for distributed data storage that uses MapReduce as a programming model for data processing and consists of the map and reduceByKey functions. In general, the map function performs a given operation on each data point, and reduceByKey function performs a summary operation, i.e., it combines the output of all the map functions. In the working of the SLFCM algorithm, the map and reduceByKey are considered as the essential functions. The map function computes the membership degree of each data point, and the reduceByKey function combines the output of all the map functions and updates the cluster centers. Figure 6.1 shows how SLFCM uses the map and reduceByKey functions for any two pairs of data points. The details of these functions are presented subsequently. The parallel computation of membership degree reduces the run-time as compared to the linear execution on a single machine. Furthermore, in Line 7 of ***Algorithm 2.2***, membership degrees of all data points are needed to update the cluster center values. Thus, Line 7 of ***Algorithm 2.2*** must be executed after membership degrees of all points have been determined. So, according to this, in ***Algorithm 6.1***, the membership degrees of all the data points are combined at the master node and stored in terms of membership information $I^{'}$, which is used in Eq. (6.2) to update the cluster

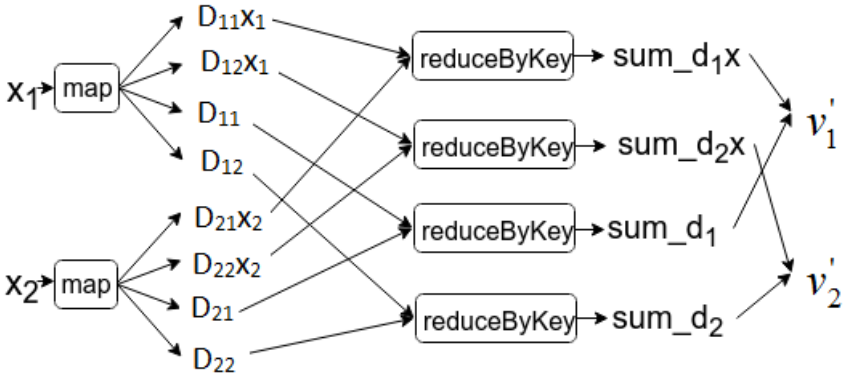


Figure 6.1: Storage space optimization for any two pairs of data points

.

center $v_j^{'}$. Then, by using Line 9 of ***Algorithm 6.1***, the difference between previously initialized cluster center and currently updated cluster center values are computed. According to Line 10 of the ***Algorithm 6.1***, this procedure is repeated until no significant change is observed in the values of cluster centers. ***Algorithm 6.1***, present the steps involved in SLFCM.

Algorithm 6.1 Algorithm for Scalable Literal Fuzzy C-Means (SLFCM) to Iteratively Minimize $J_m(U, V^{'})$

1: **Input**: X, V, c, m, ϵ ; X is a dataset consists of n data points such that $X = \{x_1, x_2, ...x_n\}$, V is the set of initial cluster centers represented as $V = \{v_1, v_2, ...v_c\}$, c denotes the number of clusters, and ϵ represents the termination criteria.

2: **Output**: $V^{'}, I^{'}$; $V^{'}$ represents set of final cluster centers and $I^{'}$ represents the membership information of all the data points.

3: **Begin**

4: **Initialize** V by randomly choosing some data points from X.

5: **Compute** membership information.

$$I' = X.map().reduceByKey() \tag{6.1}$$

6: **Compute** cluster centers.

7: **for** $< j, < sum\_d_jx, sum\_d_j >>$ in $I^{'}$ **do**

$$v_j^{'} = \frac{sum\_d_jx}{sum\_d_j}, \forall j \in [1, c] \tag{6.2}$$

8: **end for**

9: **Calculate** change in cluster center values.

$$\epsilon^{'} = \| V' - V \| \tag{6.3}$$

10: **If** $\epsilon^{'} < \epsilon$: *Stop*; **Else**: $V = V'$ and Go to Line 5.

11: **Return** $V^{'}, I^{'}$

12: **End**

Figure 6.2 depicts the workflow of SLFCM implemented on the Apache Spark cluster that takes Big Data and performs its parallel processing on Spark Cluster. In addition to the parallel processing of data in the proposed SLFCM algorithm, we make its implementation more efficient by reducing the space requirements. To do this, we avoid storing the large membership matrix of data points. For example: In ***Algorithm 2.2***, the membership matrix U is needed to compute the cluster center $v_j^{'}$ using Eq. (2.8). Instead of storing the huge membership matrix U, we directly compute the value of the parameter present in the numerator and denominator of Eq. (2.8), i.e., $[u_{ij}]^m x_i$ and $[u_{ij}]^m$ (for every point x_i and cluster center v_j) by using map function, which is represented as $D_{ij}x_i$ and D_{ij}, respectively. The computation of $D_{ij}x_i$ and D_{ij} eliminates the need of storing the huge membership matrix U. Then, we do the summation of all the $D_{ij}x_i$ values and all the D_{ij} values of the data point corresponding to cluster center v_j by using a reduceByKey function to compute the numerator and denominator of Eq. (2.8) that is represented as sum_d_jx and sum_d_j, respectively and stored in terms of membership information in a variable $I^{'}$ (as stated in Line 7 of ***Algorithm 6.1***). Then, we access these values to compute the new cluster centers using Eq. (6.2). Due to this, we save a significantly huge amount of space and reduce the computational time [111]. Figure 6.1, demonstrates the entire procedure of storage space optimization for any two pairs of data points. As discussed earlier, the proposed SLFCM algorithm works in two phases, i.e., the map phase (using map function) and reduce phase (using the reduceByKey function). So, the detailed description of these functions is presented next.

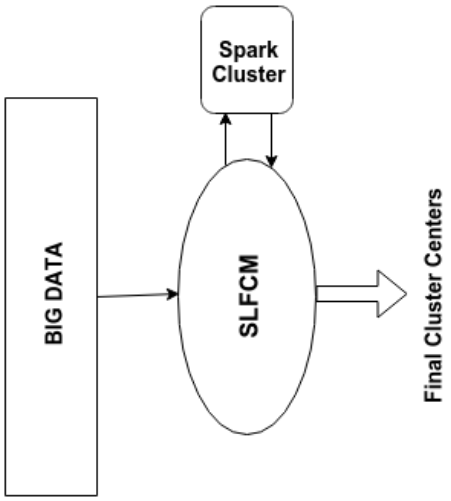


Figure 6.2: Workflow of SLFCM Algorithm.

Map function

The map function works in the following manner. First, the entire data is split into several chunks at the master node. Then, these chunks are assigned to different worker nodes for its processing across the cluster. As discussed earlier, calculation of the membership degree of each data point is independent of other data points. Therefore, we perform this operation independently for each data point on different worker nodes and combine the resulting values on the master node is equivalent to the operation performed for all the data points on a single machine. The map function defined in ***Algorithm 6.2***, calculates the membership degree of a data point corresponding to each cluster center and returns the results to RDD in terms of key/value pairs [128]. RDD is nothing but a data structure used to store the samples efficiently in memory as already discussed in section 2.6.2 (chapter 2). Here a map function is created corresponding to each data point, and it gives as many outputs equal to the number of clusters as results in RDD in terms of key-value pairs where each key represents a cluster number and a value contains the result of an operation performed in a map function. ***Algorithm 6.2***, describes the set of operations performed during a map function to obtain $[u_{ij}]^m x_i$ and $[u_{ij}]^m$ for every point x_i and cluster center v_j from chapter 2, Eq. (2.8). In this algorithm, for the sake of simplicity $[u_{ij}]^m x_i$ and $[u_{ij}]^m$ are represented as $D_{ij}x_i$ and D_{ij}, respectively. Here, the stated map function produces outputs equal to the number of clusters (j, $\forall j \in [1, c]$) and writes these outputs as results in RDD in terms of key-value pairs where each j represents a key and $D_{ij}x_i$, D_{ij} represents the corresponding values. ***Algorithm 6.2*** outlines the steps of map function.

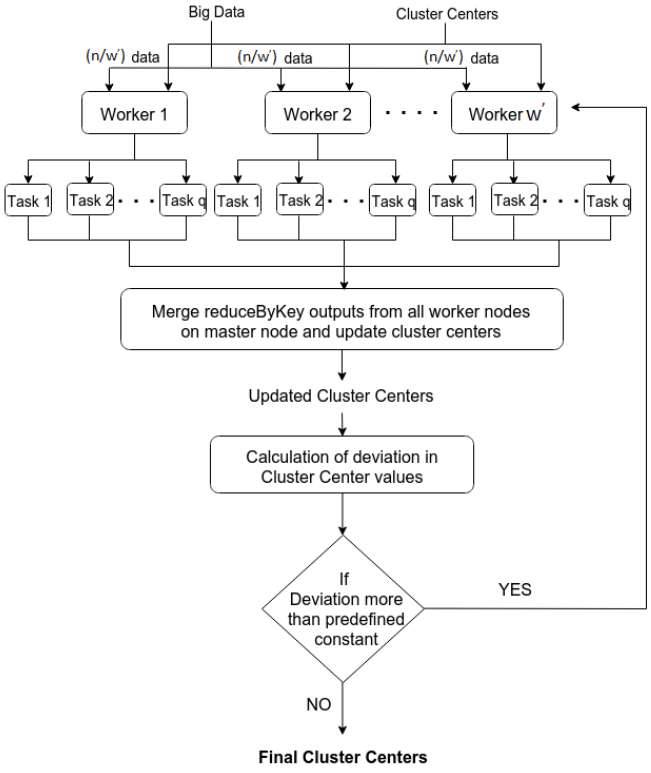
Algorithm 6.2 Algorithm for Map Function (map())

1: **Input**: x_i, V
2: **Output**: $< j, < D_{ij}x_i,\ D_{ij} >>$
3: **Begin**
4: **for each** v_j **in** V **do**
5: j = index of cluster center v; $\forall j \in [1, c]$.
6: D_{ij} = membership degree of x_i with respect to v_j.
7: $D_{ij}x_i = D_{ij} * x_i$
8: **yield** $< j, < D_{ij}x_i,\ D_{ij} >>$
9: **end for**
10: **End**

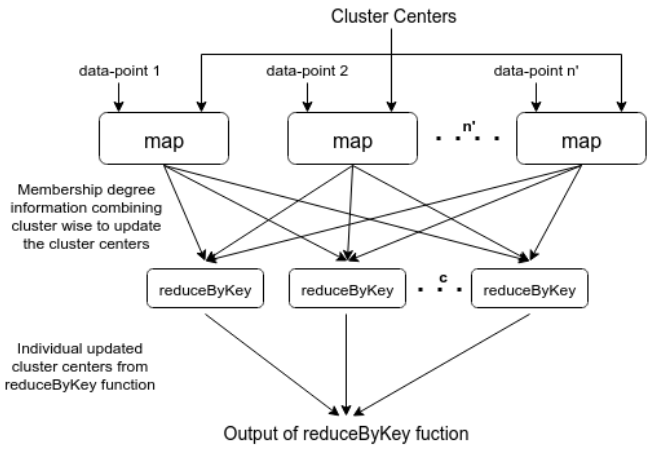
The parameter x_i represents the i^{th} data point, **yield** is used when a function

has multiple return values, and $< j, < D_{ij}x_i,\ D_{ij} >>$; $\forall j \in [1, c]$ represents key-value pairs as outputs of a map function.

Figure 6.3(a), shows the overview of working of SLFCM on the Apache Spark cluster. It describes how the n number of data points is divided and sent to $w^{'}$ worker nodes. Thus, each worker node is assigned with $(n/w^{'})$ data which is assumed to be as $n^{'}$ for the sake of simplicity. Figure 6.3(b), demonstrates the operations which are performed on a chunk of data given to a task where each task is composed of many map functions.



(a) Overview of working of SLFCM.



(b) Steps performed on a chunk of data given to a task executed on a worker node.

Figure 6.3: SLFCM Algorithm implemented on Apache Spark Cluster.

ReduceByKey function

As discussed earlier, a map function produces many outputs equal to the number of clusters formed with fuzzy clustering and returns the results in RDD in terms of many key-value pairs. Then, the outputs of map functions that have the same key (i.e., with respect to each j^{th} cluster centers) corresponding to each $D_{ij}x_i$ and D_{ij} are combined using the reduceByKey function. This function performs operations on the key-value pairs that have the same key. However, for the sake of simplicity, here we describe the operations that are performed on two such map outputs that share the same key. Spark takes care of performing the same operations on all the map outputs (key-value pairs). Thus we have calculated $[u_{ij}]^m x_i$ and $[u_{ij}]^m$ as $D_{ij}x_i$ and D_{ij}, respectively for every point x_i and j^{th} cluster center, during the map phase, now we need to add all these values to find the numerator and denominator of Eq. (2.8), presented in chapter 2 required to update the cluster center values. This is done by using the reduceByKey function which is described in ***Algorithm 6.3***. This gives us the numerator ($\sum_{i=1}^{n}[u_{ij}]^m x_i$) and the denominator ($\sum_{i=1}^{n}[u_{ij}]^m$) of Eq. (2.8) for every cluster center v_j as $sum\_d_j x$ and sum_d_j, respectively.

Algorithm 6.3 Algorithm for ReduceByKey Function (reduceByKey())

1: **Input**: aa, bb such that $aa =< j, < (D_{ij}x_i)_{aa}, (D_{ij})_{aa} >>$, $bb =< j, < (D_{ij}x_i)_{bb}, (D_{ij})_{bb} >>$.
2: **Output**: $< j, < sum\_d_j x, sum\_d_j >>$
3: **Begin**
4: $sum\_d_j x = (D_{ij}x_i)_{aa} + (D_{ij}x_i)_{bb}$
5: $sum\_d_j = (D_{ij})_{aa} + (D_{ij})_{bb}$
6: **Return** $< j, < sum\_d_j x, sum\_d_j >>$
7: **End**

The parameters aa and bb are the output of two map functions corresponding to the j^{th} cluster center, $(D_{ij}x_i)_{aa}$ and $(D_{ij}x_i)_{bb}$ denotes the value of $D_{ij}x_i$ corresponding to map function outputs aa and bb, respectively, $(D_{ij})_{aa}$ and $(D_{ij})_{bb}$ denotes the value of D_{ij} corresponding to map function outputs aa and bb, respectively. Similarly, the outputs of all the map functions are combined which share the same key. Then, the output of the reduceByKey function is used to calculate the new cluster center values using Eq. (6.2) on the master node.

6.2.2 Proposed Design of a Novel SRSIO-FCM Algorithm to Handle Big Data

The SRSIO-FCM algorithm starts by partitioning the entire data into various subsets (chunks). These subsets are created by randomly selecting data points from the entire dataset without replacement. The data points present in a subset are distinct from the data points present in other subsets. The clustering of each subset is done in parallel on the Apache Spark Cluster. ***Algorithm 6.4***, outline the steps involved in the SRSIO-FCM algorithm for clustering of each subset. For clustering of the first subset, the cluster centers are randomly initialized. SRSIO-FCM compute cluster centers, and membership information for the first subset X_1, denoted as $V^{'}$ and $I^{'}$, respectively in parallel by using Eqs. (6.1), (6.2), and (6.3). Then, it uses $V^{'}$ as initial cluster centers for clustering of the second subset. After this, the membership information and cluster centers are computed for the second subset X_2, denoted as I and $V^{'}$, respectively in parallel by again using Eqs. (6.1), (6.2), and (6.3).

But unlike RSIO-FCM, SRSIO-FCM does not use the final cluster centers of the second subset ($V^{'}$), as the initial cluster centers for clustering of the third subset. This is because SRSIO-FCM takes into account the fact that random partitioning may result in the subsets containing the data points of distinct classes. Therefore, the cluster centers of these two subsets will be significantly different from each other. So to use a better approximation of initialization of cluster centers for the clustering of any subset, SRSIO-FCM avoids using cluster centers of the previous subset as initial cluster centers for clustering of the current subset. Instead, it combines the membership information of all the processed subsets. Like, here in Eq. (6.4) the membership information of first two processed subsets denoted as $I^{'}$ and I are combined, and the new cluster centers are evaluated using the Eq. (6.2). These cluster centers provide a better estimation to the actual cluster centers since they are calculated using the combined membership information of a larger number of data points, which covers the wider sample space. This is done to avoid feeding the highly deviated cluster centers as an input for clustering any subsequent subset and thus, it helps in reducing the sudden increase in the number of iterations during clustering of a particular subset. The same procedure is followed for clustering of all the

subsequent subsets in parallel on the Apache Spark cluster. ***Algorithm 6.4***, outlines the steps of SRSIO-FCM.

Algorithm 6.4 Algorithm for Scalable Random Sampling with Iterative Optimization Fuzzy C-Means to Iteratively Minimize $J_m(U, V^{'})$

Input : X, V, c, m, ϵ
Output: $V^{'}, I^{'}$
Begin
Load X as n_s sized randomly chosen subsets $X = \{X_1, X_2 \ldots\ldots X_s\}$.
Sample X_1 from X without replacement.
$V^{'}, I^{'} = SLFCM(X_1, V, c, m)$
for $e = 2$ to s do
    $V^{'}, I = SLFCM(X_e, V^{'}, c, m)$
    Merge the partition of all processed subsets using Eq. (6.4).
    for $j = 1$ to c **do**

$$I_j^{'} =< j, < (sum\_d_j x)_{I_j}, +(sum\_d_j x)_{I_j^{'}}, (sum\_d_j)_{I_j}, +(sum\_d_j)_{I_j^{'}} >> \quad (6.4)$$

    end for
    Compute new cluster centers $v_j^{'}$ using $< j, < sum\_d_j x,\ sum\_d_j >>$ in $I^{'}$ by Eq. (6.2) $\forall j \in [1, c]$
end for
Compute the objective function using Eq. (2.6).
Return $V^{'}, I^{'}$.
End

Figure 6.4, shows the complete workflow of SRSIO-FCM. It shows that how data is randomly partitioned into various subsets and cluster centers are randomly initialized for clustering of the first subset (subset 1). It also shows that how the cluster centers, and membership information for the subset 1 are determined. Then in the successive steps, it shows the computation of cluster centers, and membership information during clustering of the second subset (subset 2). After this, the cluster centers at each step are obtained by combining membership information of all previously processed subsets. Thus, in the end, we get the cluster centers, which are obtained by combining the membership information of the entire data. Therefore, it covers a wider sample space and can produce more accurate cluster centers for each subset. To compare the SRSIO-FCM algorithm, one more algorithm rseFCM [36] is extended to handle Big Data, which are discussed next.

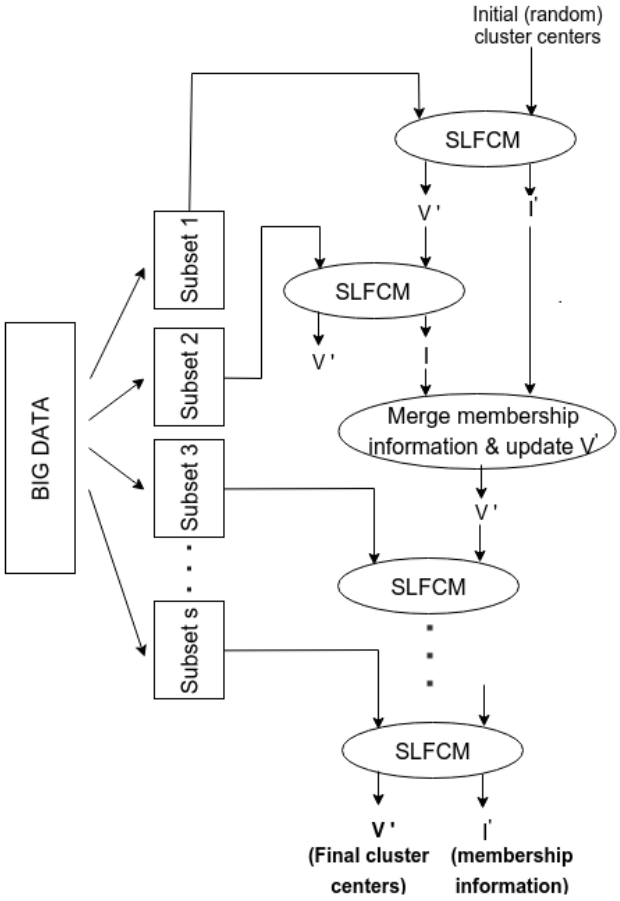


Figure 6.4: Workflow of SRSIO-FCM Algorithm.

6.2.3 Enhanced Scalable Version of rseFCM Algorithm to Handle Big Data

As discussed in section 2.5 (chapter 2), rseFCM takes a small subset of the entire data and performs clustering over it. The clustering results obtained on a small subset are used to compute clustering results of the entire data. This can be done in parallel to ensure scalability, and for better optimization of space utilization during clustering, the scalable version of this algorithm is proposed, termed as SrseFCM. In this, the map and reduceByKey functions are applied on the entire data using Eq. (6.1) and the final cluster centers are obtained using Eq. (6.2). For the computation of the final cluster centers, we do not store the large membership matrix. Instead, we compute the numerator and denominator as $sum\_d_j x$ and sum_d_j, respectively to perform the storage space optimization (discussed in section 6.2.1). This makes our implementation space efficient. ***Algorithm 6.5***, presents the step-wise procedure for SrseFCM and Figure 6.5 depicts the workflow of SrseFCM.

Algorithm 6.5 Algorithm for Scalable random sampling plus extension Fuzzy C-Means to approximately minimize $J_m(U, V^{'})$

1: **Input** : X, V, c, m, ϵ
2: **Output**: $V^{'}, I^{'}$.
3: **Begin**
4: **Sample** the n_s objects from X without replacement, denoted as X_s.
5: $U_s, V = LFCM(X_s, V, c, m)$
6: **Compute** membership information of entire data using Eq. (6.1)
7: **Compute** cluster centers $v_j^{'}$ using $< j, < sum\_d_j x,\ sum\_d_j >>$ in $I^{'}$ by Eq. (6.2), $\forall j \in [1, c]$
8: **Return** $V^{'}, I^{'}$.
9: **End**

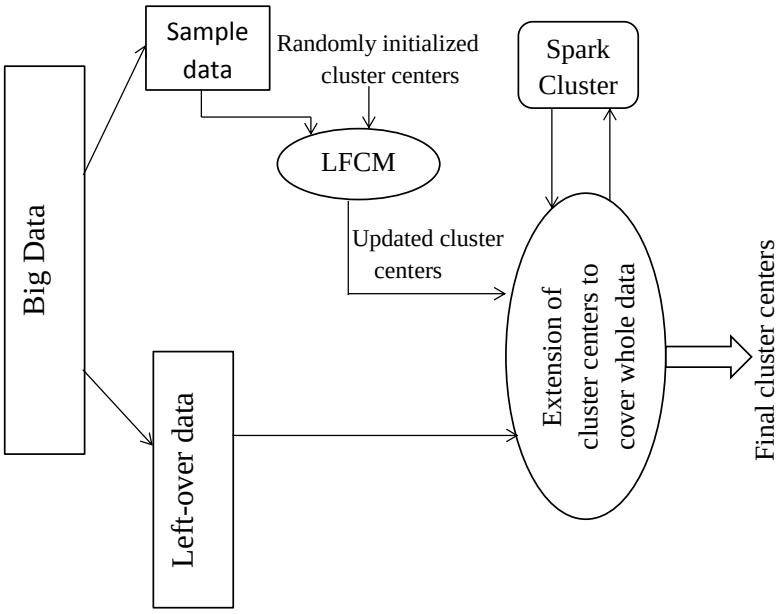


Figure 6.5: Workflow of SrseFCM Algorithm.

6.3 Complexity Analysis

In this section, the proposed scalable clustering algorithms, i.e., SLFCM, Srse-FCM, and SRSIO-FCM are analyzed in terms of space and time complexity. All operations and storage space are counted as a unit cost. The space complexity is calculated on the amount of data held in RAM during computation. Table 6.1, shows the complexities of scalable fuzzy clustering algorithms. The asymptotic estimates shown in the table indicate the growth in time and space in terms of problem variables defined as follows: X is a set that consists of n number of data points in the d-dimensional space (which are unaffected by the changes in counting procedures) such that $X \in R^d$; c is the number of clusters; $w^{'}$ is the number of worker nodes in Spark Cluster; and for SRSIO-FCM algorithm

Table 6.1: Space and time complexity of Scalable Fuzzy Clustering Algorithms.

Algorithm	Time Complexity	Space Complexity
SLFCM	$O(ncdt/w^{'})$	$O(ncd)$
SrseFCM	$O(ncd/w^{'})$	$O(ncd)$
SRSIO-FCM	$O(ncdt/w^{'})$	$O(ncd/s)$

the dataset X is partitioned by random sampling without replacement into s number of subsets represented as $X = \{X_1, X_2, ..X_s\}$. Each subset has (n/s) data points. The number of iterations required for termination is taken as t. It may differ, but for the sake of simplicity and comparison, t is taken as the maximum of the number of iterations for one run of SLFCM, SrseFCM, and SRSIO-FCM.

The complexity analysis is done as follows. Firstly, we calculate the cost of computation of membership degree during the map phase. Each map operation computes the membership degree of one data point corresponding to the c number of clusters. Since each data point has d dimensions, calculation of each membership degree takes $O(d)$ time and $O(d)$ space. Therefore, each map operation takes $O(cd)$ time and $O(cd)$ space. The reduceByKey operation linearly adds the values of all the map outputs corresponding to one cluster on each worker node and combines the resulting values on the master node. Assuming equal distribution of work to the $w^{'}$ worker nodes, each node takes $O(n^{'}d)$ time and $O(cd)$ space, where $n^{'}$ is the number of data points fed as input to one worker node. Combining the results of reduceByKey operations takes $O(cw^{'}d)$ time and $O(cd)$ space.

SLFCM performs map and reduceByKey operations on the entire dataset. Each worker node processes $n/w^{'}$ data. All the worker nodes process data in parallel. Thus, the map phase takes a total of $O(ncd/w^{'})$ time and $O(ncd)$ space for each iteration across all worker nodes. Assuming that SLFCM runs for t iterations, the total time taken for map phase is $O(ncdt/w^{'})$. Since map outputs have been held in memory only for the duration of one iteration of SLFCM, thus the total space complexity for the map phase of SLFCM is $O(ncd)$. The reduceByKey function linearly adds $n/w^{'}$ map outputs, corresponding to each cluster center in parallel, on each worker node. This takes $O(nd/w^{'})$ time and $O(cd)$ space on a worker node. Since each worker node performs tasks on $n/w^{'}$ data in a parallel manner and in the same amount of time, thus the

total time required is $O(nd/w^{'})$. Each worker node gives outputs equal to the number of clusters (c), each of dimension d, which is aggregated and added to the master node. This takes $O(cdw^{'})$ time and $O(cdw^{'})$ space. Thus, the total time complexity for the reduceByKey phase is $O(ncd)$, and space complexity is $O(cdw^{'})$. Therefore, the time complexity of SLFCM is $O(ncdt/w^{'})$ and space complexity is $O(ncd)$, where $n \gg w^{'}, c$.

SrseFCM takes a small subset of the entire data, denoted as $n^{'}$, by randomly selecting data points without replacement from the entire data and then it performs clustering over $n^{'}$ data using LFCM. So, its time complexity would be the same as that of LFCM for ($n^{'}$) data where $n^{'} << n$. But, while extending the cluster centers to the entire data, i.e., to compute the full fuzzy partition of the entire data, membership degree of all the points are calculated. The time complexity for this final step is $O(ncd/w^{'})$, which is much higher than the time complexity of running LFCM over ($n^{'}$) data. Similarly, the space complexity of running LFCM over ($n^{'}$) data is very less as compared to the space complexity of the final extension step. In the final extension step, we compute the membership degree of n size data consists of d dimensions for c number of clusters. This extension step results in a space complexity of $O(ncd)$. Thus, the time complexity is $O(ncd/w^{'})$, and space complexity is $O(ncd)$.

SRSIO-FCM divides the entire data X into s equal subsets such that $X = \{X_1, X_2, \ldots X_s\}$, where each subset is of size (n/s). It performs scalable fuzzy clustering over each of these subsets serially. The time and space complexity of performing scalable fuzzy clustering over each subset are $O(ncdt/sw^{'})$ and $O(ncd/s)$, respectively. Since, all the subsets are processed one after another, and the resulting time complexity is $O(ncdt/w^{'})$ while the space complexity remains $O(ncd/s)$ because data corresponding to one subset is not retained in the memory while processing the next subset.

Table 6.1, shows that SLFCM and SRSIO-FCM share the same time complexity. This may lead one to think that both have the same run-time. However, this is not the case. In SRSIO-FCM, we divide the entire data into various subsets and perform clustering over each subset. So, clustering carried out by the SRSIO-FCM on each subset converge by taking the less number of iterations (t) for each subset. Therefore, SRSIO-FCM has lesser run-time, since it performs clustering on a lesser amount of data as compared to SLFCM, which performs

clustering of the entire data. As we can see in section 6.4.2, there is a significant difference in the run-time of the two algorithms. The SrseFCM algorithm uses the cluster centers that are returned after clustering of a subset of data to produce full data partitions. Although, in SrseFCM we have not estimated complexity for the completion step.

The space complexity of SRSIO-FCM is less when compared with SrseFCM and SLFCM, and is proportional to the number of data points that are present in each subset, i.e., n/s. The space complexity of SRSIO-FCM will be same as SrseFCM if the size of the subset in SRSIO-FCM is taken to be equal to the size of the subset in SrseFCM.

6.4 Experimental Evaluation

In this section, we present the experiments of our proposed scalable clustering algorithms on various Big Datasets and then a comparison of the performance of SRSIO-FCM with SLFCM and SrseFCM on different measures are presented. The measures are NMI [46], F-measure [45], ARI [47, 48], objective function, speedup, sizeup, and scaleup [49, 50]. In addition to this, a trade-off between performance gain versus accuracy is also observed for the in-depth analysis of loss of accuracy (in terms of NMI) versus a performance gain (in terms of run-time). The aim is to achieve the higher value of NMI, F-measure, and ARI and lower value of the objective function for SRSIO-FCM as compared to SrseFCM and lower run-time as compared to SLFCM which is evaluated in terms of speedup, sizeup, and scaleup. The detailed description of these measures is given in the section 2.8.1. The information about the datasets and experimental settings are discussed next.

6.4.1 Datasets and Experimental Settings

To demonstrate the efficacy of proposed SRSIO-FCM over the proposed scalable version of LFCM and rseFCM, we have created Big Datasets by taking data from various sources [154–157] and evaluate the performance of these approaches on various Big Datasets, i.e., Minst8m, Replicated-USPS, Monarch-Skin, and SUSY. The details of these datasets are presented in section 2.9, chapter 2. To evaluate the performance of all the approaches, we fix the value of fuzzifier

$m = 1.75$ and termination criteria $\epsilon = 0.001$ for these datasets used in the experimental study. Also, we have fixed the value of $c = 10$, $c = 10$, $c = 2$, and $c = 3$ for Minst8m, Replicated-USPS, Monarch-Skin, and SUSY datasets, respectively. After exhaustive experimentation, we found that these values are more suitable for the datasets because on these values the datasets achieve better performance. Also, these values are proven to work well for most of the datasets [36, 45, 167]. The specification of the parameter value for each dataset used in the experimental evaluation is given in Table 6.2. All the approaches are implemented on the Apache Spark Cluster. The experiments have been done on Spark Cluster with 3 nodes. Each node has the following configuration: Intel(R) Xenon(R) CPU E5-1607 v3 @ 3.10GHz x 4, 32GB RAM and 2TB storage. The algorithms have been implemented in Python and tested over Hadoop version 2.4 with Apache Spark version 1.4.0. As discussed earlier in the section 2.6.2, chapter 2, Apache Spark requires a cluster manager and a distributed storage system so here we used YARN [132] for cluster management and HDFS [134] for storing data across the Spark Cluster.

6.4.2 Experimental Results and Discussion

This section presents the experimental results conducted to demonstrate the effectiveness of SRSIO-FCM in comparison with SLFCM and SrseFCM on four datasets in terms of NMI, F-measure, ARI, objective function, speedup, sizeup, and scaleup. Furthermore, observations for a trade-off between performance gain versus accuracy is also conducted in terms of percentage change in run-time and NMI.

For each dataset, we compare the performance of SRSIO-FCM with SLFCM and SrseFCM algorithms using the value of the parameters presented in Table 6.2. The comparison of the values of NMI, F-measure, ARI, and Objective

Table 6.2: Specification of parameters for various datasets.

Parameters	**Datasets**			
	MNIST8m	Replicated-USPS	Monarch-Skin	SUSY
ϵ	0.001	0.001	0.001	0.001
m	1.75	1.75	1.75	1.75
c	10	10	2	3

function of SLFCM and SrseFCM with various chunk sizes of SRSIO-FCM on four datasets are presented in Tables 6.3, 6.4, 6.5, and 6.6, respectively. The results reported on the chunk sizes of the SRSIO-FCM algorithm is different from the chunk size of SrseFCM and SLFCM algorithms. This is because the SLFCM algorithm performs clustering of the entire data by processing the data in parallel on various worker nodes. Therefore, the chunk size of SLFCM is taken as 100%. On the other hand, the chunk size of the SrseFCM algorithm is taken as 0.001% because the SrseFCM works on a small subset of the entire data to produce the clustering results of the entire data. Thus, the SrseFCM works on a small chunk size of the dataset. After exhaustive experimental evaluation, we have found the chunk size (0.001%) of the SrseFCM algorithm that gives the best clustering results (with a lesser overhead in terms of run-time) and which is somewhat closer to the results achieved by SLFCM and SRSIO-FCM algorithms. Therefore, for the purpose of fair comparison, we have taken the chunk size of SrseFCM as 0.001%. The chunk sizes of the SRSIO-FCM reported here is higher than the chunk of SrseFCM because the overhead (in terms of run-time) increases with the same clustering results in the lesser chunk size. Furthermore, Table 6.7 compares the performance of SRSIO-FCM with SLFCM in terms of speedup. Figure 6.6 displays a comparison of the run-time of SLFCM with various chunk sizes of SRSIO-FCM in terms of sizeup. Figure 6.7 investigate the scalability of SRSIO-FCM as compared to SLFCM for different datasets. Figure 6.8 demonstrates the trade-off between performance gain versus accuracy in terms of percentage change in run-time and NMI measure.

Table 6.3, tabulates the value of NMI measure computed on four datasets using SLFCM, SrseFCM and with various chunk sizes of SRSIO-FCM. NMI indicates the quality of clustering. The higher NMI value indicates better clustering results. The table shows that the value of NMI measure for SLFCM and SRSIO-FCM is very close whereas, in most of the cases, SrseFCM attains a much lower value of the NMI measure as compared to that of SLFCM and SRSIO-FCM. In addition to this for some datasets, there is no significant variation in the value of NMI for various chunk sizes of SRSIO-FCM. Thus, we can conclude that SRSIO-FCM with the given chunk sizes performs better than SrseFCM and equally well as SLFCM in terms of NMI measure.

Table 6.4, tabulates the value of F-measure computed on four datasets using

Table 6.3: NMI for SLFCM, SrseFCM, and SRSIO-FCM with various chunk sizes on different datasets.

Algorithm	**Chunk size**	**Datasets**			
		MNIST8m	Replicated-USPS	Monarch-Skin	SUSY
SRSIO-FCM	1%	0.117094	0.365415	0.012703	0.028723
	2.5%	0.125052	0.365966	0.012703	0.029032
	5%	0.122007	0.365119	0.012703	0.028674
	10%	0.118471	0.365557	0.012705	0.028667
	20%	0.126027	0.365292	0.012703	0.028798
SLFCM	100%	0.122540	0.396155	0.012703	0.025895
SrseFCM	0.001%	0.118607	0.253252	0.013281	0.001194

SLFCM, SrseFCM and with various chunk sizes of SRSIO-FCM. F-measure indicates the quality of clustering. Thus, a higher value of F-measure indicates better clustering. The table shows that the values of F-measure for SLFCM and SRSIO-FCM is very close whereas, in most of the cases, SrseFCM attains a significantly lower of the F-measure as compared that of SLFCM and SRSIO-FCM. Also, the values of F-measure for various chunk sizes of SRSIO-FCM is approximately equal. Thus, we can conclude that SRSIO-FCM performs as good as SLFCM and better than SrseFCM in terms of F-measure.

Table 6.5, shows the value of ARI measure computed on four datasets using SLFCM, SrseFCM and with various chunk sizes of SRSIO-FCM. The ARI value compares the clustering result with the ground truth labels. The higher value of ARI measure indicates better clustering results. The table shows that the values of ARI measure for SLFCM and SRSIO-FCM on MINST8m, Monarch-

Table 6.4: F-measure for SLFCM, SrseFCM, and SRSIO-FCM with various chunk sizes on different datasets.

Algorithm	**Chunk size**	**Datasets**			
		MNIST8m	Replicated-USPS	Monarch-Skin	SUSY
SRSIO-FCM	1%	0.362525	0.787940	0.648496	0.641601
	2.5%	0.365356	0.788448	0.648497	0.642083
	5%	0.363194	0.788653	0.648496	0.640909
	10%	0.364882	0.788539	0.648508	0.641277
	20%	0.379519	0.788608	0.648497	0.641660
SLFCM	100%	0.381972	0.842952	0.648497	0.607977
SrseFCM	0.001%	0.380723	0.565469	0.619401	0.453477

Table 6.5: ARI for SLFCM, SrseFCM, and SRSIO-FCM with various chunk sizes on different datasets.

Algorithm	**Chunk size**	**Datasets**			
		MNIST8m	Replicated-USPS	Monarch-Skin	SUSY
SRSIO-FCM	1%	0.048867	0.215461	0.035881	0.042860
	2.5%	0.052707	0.215343	0.035882	0.043153
	5%	0.051061	0.215777	0.035881	0.042747
	10%	0.049511	0.215533	0.035889	0.043010
	20%	0.053534	0.215607	0.035883	0.042879
SLFCM	100%	0.052189	0.268365	0.035881	0.038094
SrseFCM	0.001%	0.049464	0.294295	0.022201	0.0314584

Skin, and Susy datasets are very close and very low for the SrseFCM. In the case of Replicated-USPS dataset, the ARI values of SRSIO-FCM on different chunk sizes are approximately 20% below those of SLFCM approach. This is due to the repetition of data points in Replicated-USPS dataset. In this dataset, the proportion of data points belonging to a class is significantly higher than data points belonging to other classes. As discussed earlier, the SRSIO-FCM performs clustering of the entire data by partitioning it into various subsets. Therefore, it is possible that the randomly chosen subsets contain more data points belonging to the majority class. This results in the generation of skewed cluster centers in comparison of the SLFCM which perform clustering of the entire data at once. Therefore, ARI values are 20% below as compared to the SLFCM. In general, for all the datasets the value of ARI measure for various chunk sizes of the SRSIO-FCM is approximately equal. Thus, it can be referred from the results that the SRSIO-FCM performs as good as the SLFCM in terms of ARI measure. Furthermore, it performs approximately equally well for all the chunk sizes.

Table 6.6, displays the value of objective function computed on four datasets by applying our proposed algorithms, i.e., SLFCM, SrseFCM, and SRSIO-FCM with various chunk sizes of 1%, 2.5%, 5%, 10% and 20%. The objective function values for SLFCM and SRSIO-FCM is found to be very close, whereas the objective function values for the SrseFCM is much higher than both the SLFCM and SRSIO-FCM. Moreover, the values of an objective function for various chunk sizes of the SRSIO-FCM is approximately equal. Thus, we may conclude that the SRSIO-FCM minimizes the objective function equally well as the SLFCM

Table 6.6: Objective Function value for SLFCM, SrseFCM, and SRSIO-FCM with various chunk sizes on different datasets.

Algorithm	**Chunk size**	**Datasets**			
		MNIST8m	Replicated-USPS	Monarch-Skin	SUSY
SRSIO-FCM	1%	74556041.35	83800629	9.53E+016	24522339.61
	2.5%	74556041.52	83800623	9.53E+016	24522336.73
	5%	74556038.97	83800617	9.53E+016	24522337.77
	10%	74556042.40	83800622	9.53E+016	24522346.13
	20%	74556036.17	83800622	9.53E+016	24522338.22
SLFCM	100%	74556031.67	83800498	9.53E+016	24522337.10
SrseFCM	0.001%	74647034.95	173762000	1.52E+17	24556967.02

and better than the SrseFCM in almost all the cases.

Table 6.7, shows the speedup evaluation of the SRSIO-FCM and SLFCM on different datasets by varying the number of nodes (systems) in the cluster and keeping the chunk size constant. The number of nodes in the cluster is varied from 1 to 3. We varied the chunk size of datasets from 1% to 20% to compare the run-time of the SRSIO-FCM with SLFCM which uses 100% data. The results show that the SRSIO-FCM takes a very less run-time in comparison with the SLFCM. Thus, SRSIO-FCM achieves an excellent speedup performance compared to the SLFCM irrespective of the chunk size. Hence, the SRSIO-FCM performs better than the SLFCM and can efficiently cluster Big Data.

Figure 6.6, shows the sizeup evaluation of the SRSIO-FCM and the SLFCM on different datasets by varying the chunk size of the datasets and keeping the number of nodes in the cluster constant. The run-time of the SRSIO-FCM is evaluated with the varying chunk size of datasets from 1% to 20% in comparison with the SLFCM. The results show that the run-time of SLFCM on a different number of nodes in a cluster are comparatively much higher than the run-time achieved by the SRSIO-FCM. Specifically, with 3 nodes, the sizeup performance of the SRSIO-FCM is very good for all the datasets. Thus, the SRSIO-FCM achieves the good sizeup performance in comparison with the SLFCM. Hence, it is inferred that the SRSIO-FCM can handle Big Data efficiently.

As shown in Figure 6.7, scaleup analysis is reported to evaluate the performance of SRSIO-FCM in comparison with the SLFCM algorithm. To demonstrate how well the SRSIO-FCM algorithm handle Big datasets compared to the SLFCM algorithm when more nodes (systems) are available, we have performed

Table 6.7: Speedup Analysis for SLFCM and SRSIO-FCM.

(a) Run-time comparison (in seconds) of MINST8m data.

Number of nodes	**Algorithms**					
	SRSIO-FCM					**SLFCM**
	Chunk Size					Chunk Size
	1%	2.5%	5%	10%	20%	100%
1 Node	2704	1848	1488	1484	1683	3987
2 Nodes	1456	1056	1008	1064	1173	2700
3 Nodes	972	616	528	588	663	2294

(b) Run-time comparison (in seconds) of Replicated-USPS.

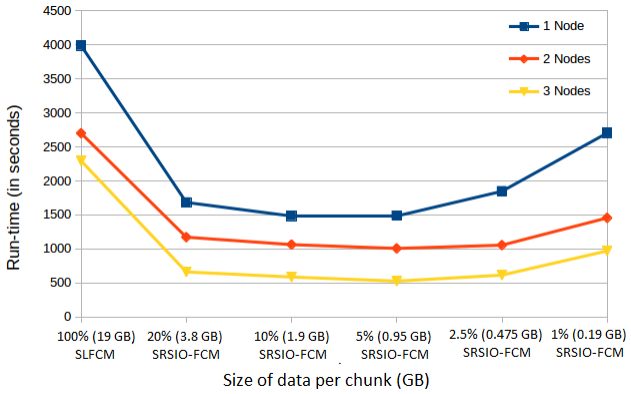
Number of nodes	**Algorithms**					
	SRSIO-FCM					**SLFCM**
	Chunk Size					Chunk Size
	1%	2.5%	5%	10%	20%	100%
1 Node	26237.83	14810	16035.55	27198.31	45868.00	199228.73
2 Nodes	6984.67	5386.99	7389.05	14252.91	29381.47	103426.81
3 Nodes	3140	2700	3179	5568	10746	25537.92

(c) Run-time comparison (in seconds) of Monarch-Skin data.

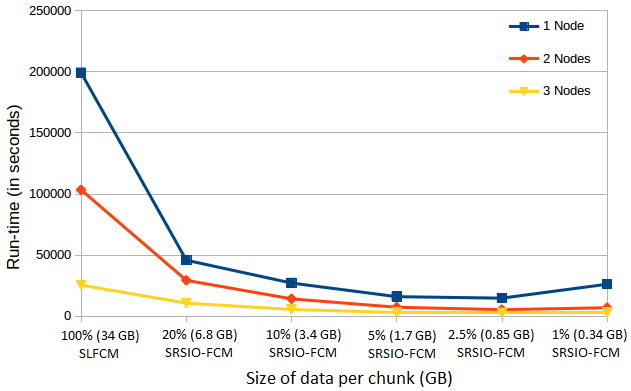
Number of nodes	**Algorithms**					
	SRSIO-FCM					**SLFCM**
	Chunk Size					Chunk Size
	1%	2.5%	5%	10%	20%	100%
1 Node	301072.07	261984.09	259540.11	240017.01	270691.01	600103.17
2 Nodes	202200	186116	188799	184500	187616	401305.81
3 Nodes	74140	69552	64890	62100	63960	117226.04

(d) Run-time comparison (in seconds) of SUSY data.

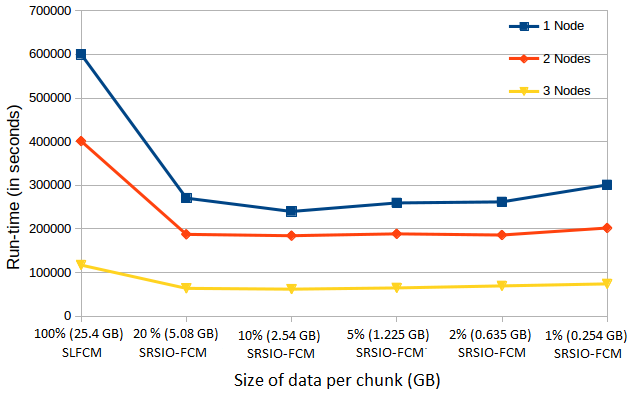
Number of nodes	**Algorithms**					
	SRSIO-FCM					**SLFCM**
	Chunk Size					Chunk Size
	1%	2.5%	5%	10%	20%	100%
1 Node	1158.16	772.71	616.67	738.82	805.85	3020.83
2 Nodes	296.30	200.66	211.27	294.67	492.29	2133.54
3 Nodes	228	112	128	188	200	864.04



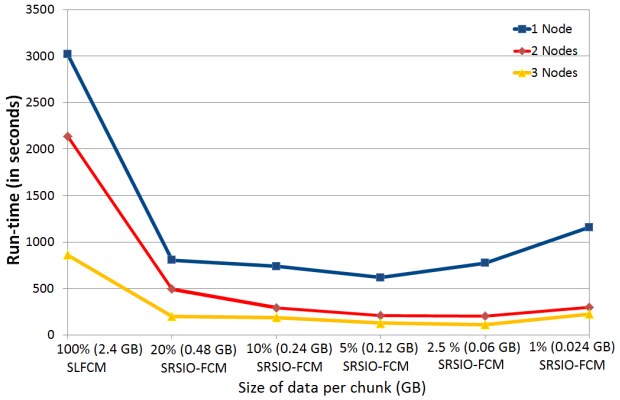
(a) Run-time comparison on MNIST8m data.



(b) Run-time comparison on Replicated-USPS data.



(c) Run-time comparison on Monarch-Skin data.



(d) Run-time comparison on SUSY data.

Figure 6.6: Sizeup Analysis for SLFCM and SRSIO-FCM.

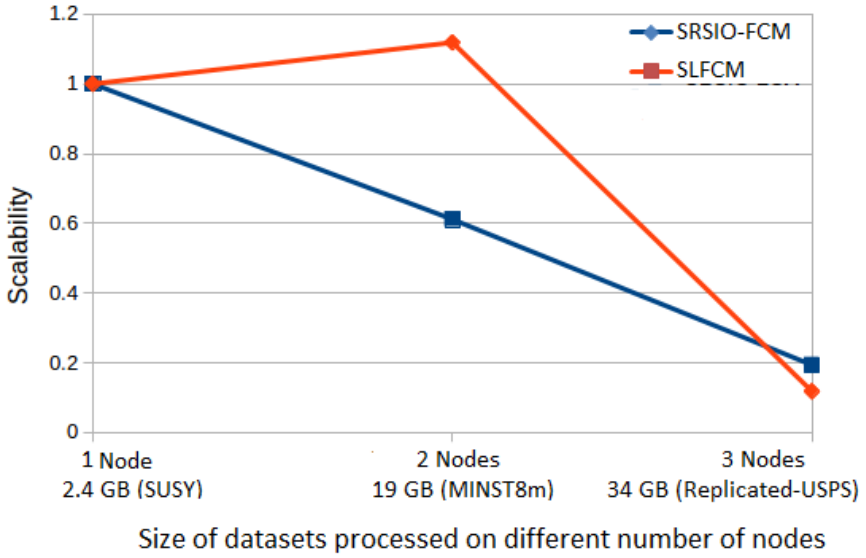


Figure 6.7: Scaleup Analysis for SLFCM and SRSIO-FCM.

the scaleup experiments where we have used the datasets of varying sizes in proportion to the number of nodes in the cluster. As discussed earlier, we are working with three nodes in the Apache Spark Cluster, so we have taken three datasets to evaluate the scalability, i.e., one dataset per node. The reported results show that the SRSIO-FCM shows decent scaleup performance over datasets of varying sizes in comparison with the SLFCM. Thus, the SRSIO-FCM scales very well and handle Big Data efficiently.

Figure 6.8, compares the performance gain versus accuracy in terms of the percentage change in run-time and the loss of accuracy (in terms of NMI) for 5% chunk size of the SRSIO-FCM as compared to the SLFCM. The results are analyzed and reported for 3 nodes in a cluster. As depicted clearly by the graph, the loss of accuracy is almost negligible as compared to the enhancement in run-time.

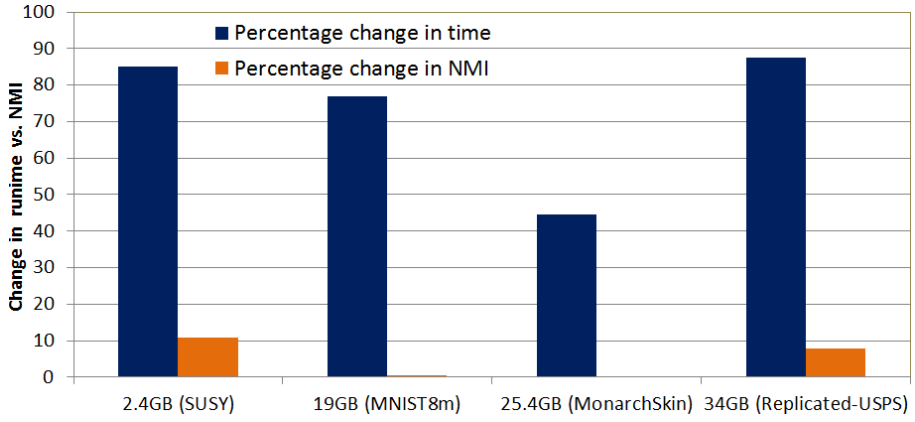


Figure 6.8: Trade-off between performance gain versus accuracy.

6.5 Summary

In this chapter, a novel scalable clustering algorithm is proposed referred to as the SRSIO-FCM algorithm for handling Big Data. The SRSIO-FCM algorithm is designed by enhancing the RSIO-FCM algorithm discussed earlier. The SRSIO-FCM partitioned the Big Data into various chunks, and processed the data points present within the chunk in a parallel manner. One distinctive characteristic of the SRSIO-FCM is that due to the parallel processing of data points within the chunk, it achieves a significant reduction in run-time for the clustering of such huge amount of data, without compromising the quality of clustering results. The other important characteristic is that during the execution of the proposed algorithm, we eliminate the need for storing the large membership matrix, which significantly reduces the run-time and storage space and thus, it makes the execution of the proposed algorithm much faster. This is a good optimization strategy for clustering of Big Data since the membership matrix is too huge to be stored. The SRSIO-FCM algorithm is implemented using the Big Data processing framework called Apache Spark to deal with the challenges associated with fuzzy clustering for handling Big Data.

To compare our proposed SRSIO-FCM, two other basic algorithms, i.e., rseFCM and LFCM are enhanced to make these algorithms scalable for handling Big Data. To do this, we implemented the LFCM and rseFCM on the Apache Spark Cluster and designed their scalable versions which are termed as SrseFCM and SLFCM. Experimental results are evaluated on several Big Datasets in terms of various performance measures such as NMI, F-measure, ARI, objective function, speedup, sizeup, and scaleup. Moreover, a trade-off between performance gain versus accuracy is also reported for in-depth analysis of experimental results. The efficacy of the SRSIO-FCM is analyzed in terms of space and time complexity. It is observed that the space complexity of the SRSIO-FCM is very less in comparison with the SLFCM and SrseFCM both of which share the same space complexity. The SLFCM and SRSIO-FCM share the same time complexity, but the run-time of SRSIO-FCM is significantly lesser as compared to the SLFCM. This is because SRSIO-FCM performs clustering on a lesser amount of data by dividing the entire data into various subsets that lead to faster convergence by taking the less number of iterations as compared to the SLFCM

which performs clustering on the entire data. The same can be verified with the experimental evaluations carried out on several Big Datasets. The empirical evaluations show that the SRSIO-FCM significantly outperformed over the SLFCM and SrseFCM by achieving higher or comparable clustering results in terms of NMI, F-measure, ARI, and objective function. Also, the SRSIO-FCM achieves a significant reduction in run-time in terms of speedup, sizeup, scaleup as compared to the SLFCM. Furthermore, a trade-off between performance gain versus accuracy is also evaluated, which shows that the loss of accuracy is almost negligible as compared to the reduction in run-time. The results indicate that the SRSIO-FCM has great potential in Big Data clustering.

Chapter 7

Application of Scalable Clustering Algorithm along with a Novel Feature Extraction Approach for Classification of Protein Data

Bioinformatics has emerged as a forefront research area in Big Data analysis as a huge volume of protein sequences of complex plant genome is generated on a daily basis through various genome-sequencing projects. This kind of data is also generated on a regular basis by the DSR, Indore. Thus, an enormous amount of genome data (protein sequence database) generated by the DSR has been collected to test the performance of the proposed SRSIO-FCM algorithm discussed in the previous chapter. First of all, this genome database, which contains a huge volume of variable length protein sequences is required to be preprocessed efficiently. There exist many approaches for preprocessing of protein sequence database. Swati and Santanu [140] adopted an AMOGA to optimize the structure of RBFN. In this, a 2-gram sequence encoding method [141] was utilized to reduce the size of features. Wang et al. [142] proposed a new approach by using the 2-gram method [141] and 6-letter exchange groups [150] for extracting features from a protein sequence. The extracted features are used as an input to the Bayesian network to classify protein sequences into superfamilies. Bandyopad-

hyay [144] proposed a feature encoding method by using the 1-gram technique. The extracted features reflected the probability of occurrences of amino acids in a different position of sequences, which is used as an input to the nearest neighbor algorithm to classify protein sequences into superfamilies. Mansoori et al. [145] proposed a new feature extraction technique for extracting relevant features from protein sequences by counting the occurrence probabilities of six exchange groups [150] in each sequence. Then, it considers some interpretable fuzzy rules for assigning protein sequences into appropriate superfamilies.

Another feature extraction approach is proposed by Mansoori et al. [146], which extracts features from a protein sequence using 2 grams and a 2-gram exchange group from a dataset. It uses distance-based feature ranking method for the selection of best and most appropriate features, and then it employs a steady state genetic algorithm for extracting fuzzy rules from data that is used for the classification of protein sequences into superfamilies. Iqbal et al. [149] proposed a statistical metric-based feature selection technique for selection of discriminant and invariant features from protein sequences. It extracts different subsets of features from the original feature space using n-gram descriptors frequency-based encoding method and selects the best feature subset that classifies protein sequences into superfamilies with maximum accuracy. Swati et al. [168] applied the concept of AMOGA to optimize the structure of RBFN. In this, the most parsimonious network obtained from the Pareto front applies to the classification of protein sequences to superfamilies. Bharti and Nagamma [169] proposed a novel semi-supervised support vector machine classifier, which works with a combination of labeled and unlabeled dataset and used PCA for reducing features of protein sequence and classify protein sequences into superfamilies with high accuracy. In spite of the wide popularity and the existence of many feature extraction approaches, there is still need to develop a highly accurate and efficient feature extraction approach for the interpretation of a huge volume of variable length protein sequences. For this reason, we propose a novel CPSF approach. The advantage of the proposed method is that it represents each variable length protein sequence of huge size with a fixed-length numeric vector even if the protein sequence has a maximum sequence length of 5000. In addition to this, it considers all possible position-specific variations of amino acids in a protein sequence as well as within a superfamily.

The proposed CPSF approach is applied to a huge volume of variable length protein sequences for feature extraction, and then it is applied to a real-life genome database. This database is collected from the DSR, Indore. The proposed CPSF approach is being implemented in this genome database for its preprocessing to represent all the protein sequences present in this database in terms of fixed-length numeric feature vectors. Then, the SRSIO-FCM algorithm is applied on this preprocessed database for its categorization into various superfamilies. For this, initially the SRSIO-FCM algorithm partitioned the entire protein database into various clusters by considering the similarity among the protein sequences and then these formed clusters are used further to predict the superfamilies of unknown protein sequences. The categorization of massive protein database into various superfamilies assists the DSR scientists to enhance the productivity of next generation protein sequences of different species of plant. The detailed description of the proposed CPSF approach is presented next.

7.1 Proposed Novel Feature Extraction Approach for Protein Classification

This section presents the detailed description of the proposed CPSF approach. The CPSF approach captures the statistical characteristics of protein sequence along with positional information of amino acids to produce a fixed-length numeric feature vector for each protein sequence. The proposed CPSF approach extracts features of protein database in three stages: encoding of protein sequences, global similarity measure, and local similarity measure. In the first stage, it performs encoding of protein sequences where each protein sequence is represented in terms of exchange groups [150]. In the second stage, it computes the global similarity measure that takes into account the probability of occurrence of each amino acid in a particular position concerning the total number of protein sequences present in a particular superfamily. In the third stage, we compute the local similarity measure, which calculates the weight of each amino acid with respect to each protein sequence by considering the global similarity measure. In this way, the CPSF approach represents each protein sequence by a fixed-length numeric vector consisting of only six dimensions with probabilistic

significance. After this, extracted features are passed as an input to train the well-known classifiers such as SVM [170], NB [171], k-NN [172], and BLTA [173] so that these classifiers can classify unknown protein sequences to the respective superfamilies. The working of each stage in CPSF approach is presented next.

7.1.1 Stage One: Encoding of Protein Sequences

At the very first stage of CPSF approach, each protein sequence is encoded and represented in terms of six exchange groups. According to Dayhoff and Schwartz [150], the amino acids present in a protein sequence belongs to six exchange groups. This is because, within each exchange group, these amino acids have high evolutionary similarity. The exchange groups are effective equivalence classes of amino acids, which is formally represented as $\{e_1, e_2, e_3, e_4, e_5, e_6\}$, where e_1=$\{H, R, K\}$, e_2=$\{D, E, N, Q\}$, e_3=$\{C\}$, e_4=$\{S, T, P, A, G\}$, e_5=$\{M, I, L, V\}$ and e_6=$\{F, Y, W\}$ [142]. The protein sequences belong to different superfamilies, and within each superfamily, protein sequences share some structural similarity with each other. Here an illustration is presented by considering an example of five related protein sequences shown in Figure 7.1. These protein sequences belong to a particular superfamily, and within a superfamily, these sequences have high evolutionary similarity.

These protein sequences are encoded into six exchange groups as follows: SEQ_1, i.e. MDGNPLGN encoded as $\{M, L\} \in e_5$, $\{D, N\} \in e_2$, $\{P, G\} \in e_4$. Similarly, SEQ_2, i.e. MEGNDLQN is encoded as $\{M, L\} \in e_5$, $\{D, E, N, Q\} \in e_2$, $\{G\} \in e_4$. In the same way SEQ_3, i.e. TNQDFVRL is encoded as $\{R\} \in e_1$, $\{N, Q, D\} \in e_2$, $\{T\} \in e_4$, $\{V, L\} \in e_5$, $\{F\} \in e_6$. Similarly SEQ_4, i.e. VDQNPVEL is encoded as $\{D, Q, E, N\} \in e_2$, $\{P\} \in e_4$, $\{V, L\} \in e_5$. In the same manner SEQ_5, i.e. TEGNPIRN is encoded as $\{R\} \in e_1$, $\{E, N\} \in e_2$, $\{P, T, G\} \in e_4$, $\{I\} \in e_5$. Table 7.1 presents the encoded positional information of all the protein sequences given in Figure 7.1. Once, the protein sequences are

SEQ_1 :	M	D	G	N	P	L	G	N
SEQ_2 :	M	E	G	N	D	L	Q	N
SEQ_3 :	T	N	Q	D	F	V	R	L
SEQ_4 :	V	D	Q	N	P	V	E	L
SEQ_5 :	T	E	G	N	P	I	R	N

Figure 7.1: Primary structure of five related protein sequences in a superfamily.

Table 7.1: Encoded positional representation of amino acids.

Sequence	Positions							
	1	2	3	4	5	6	7	8
1	e_5	e_2	e_4	e_2	e_4	e_5	e_4	e_2
2	e_5	e_2	e_4	e_2	e_2	e_5	e_2	e_2
3	e_4	e_2	e_2	e_2	e_6	e_5	e_1	e_5
4	e_5	e_2	e_2	e_2	e_4	e_5	e_2	e_5
5	e_4	e_2	e_4	e_2	e_4	e_5	e_1	e_2

encoded in terms of exchange groups (as shown in Table 7.1), the global similarity between the encoded exchange groups are computed. Detailed description about this is presented next.

7.1.2 Stage Two: Global Similarity Measure

In the second stage of CPSF approach, we compute a global similarity measure by evaluating the probability of occurrences of all exchange groups at each position concerning the total number of protein sequences present in the superfamily. The global similarity measure is computed as follows:

$$(Probability)_{ab} = (Occurence)_{ab}/\eta \tag{7.1}$$

where $(Probability)_{ab}$ represents the probability of occurrence of the a^{th} exchange group at b^{th} position, $(Occurence)_{ab}$ denotes the frequency at which the a^{th} exchange group occur at b^{th} position and η represents the total number of sequences in a particular superfamily. Now, the global similarity measure is computed corresponding to the exchange groups given in Table 7.1. In this table, exchange group e_5 occurs at first position three times out of five sequences, therefore $(Probability)_{51}=\frac{3}{5}$. Similarly, the probability of other exchange groups present in this table is also computed. Thus, in Table 7.2, we have reported the values obtained after computing a global similarity measure corresponding to all the encoded sequences shown in Table 7.1. After this, the local similarity measure is calculated which determines the position specific weight of each exchange group. The detailed description of the same is presented next.

Table 7.2: Global Similarity Measure of encoded protein sequences.

Exchange groups	Positions							
	1	2	3	4	5	6	7	8
e_1	0	0	0	0	0	0	0.4	0
e_2	0	1	0.4	1	0.2	0	0.4	0.6
e_3	0	0	0	0	0	0	0	0
e_4	0.4	0	0.6	0	0.6	0	0.2	0
e_5	0.6	0	0	0	0	1	0	0.4
e_6	0	0	0	0	0.2	0	0	0

7.1.3 Stage Three: Local Similarity Measure

In the third stage of CPSF approach, the local similarity measure is computed, which determines the position specific weight of each exchange group within the sequence in terms of weight factors. These weight factor finally represent a feature vector for each protein sequence. The weight of each exchange group is computed as follows:

$$(Weight)_a^{SEQ_o} = \sum_{b=1}^{b'} (Probability)_{ab} \times (PW)_{ab}^{SEQ_o} \tag{7.2}$$

where $(Weight)_a^{SEQ_o}$ denotes the weight of a^{th} exchange group corresponding to the o^{th} protein sequence, $(Probability)_{ab}$ represents the probability of occurrence of the a^{th} exchange group at b^{th} position and $(PW)_{ab}^{SEQ_o}$ is the positional weight assign to the a^{th} exchange group based on the presence of o^{th} protein sequence at b^{th} position. The weight of exchange groups, i.e., the first encoded protein sequence ($e_5e_2e_4e_2e_4e_5e_4e_2$) present in Table 7.1 is calculated as follows:

$$(Weight)_{e_2}^{SEQ_1} = 1 \times 1 + 1 \times 1 + 0.6 \times 1 = 2.6$$
$$(Weight)_{e_4}^{SEQ_1} = 0.6 \times 1 + 0.6 \times 1 + 0.2 \times 1 = 1.4$$
$$(Weight)_{e_5}^{SEQ_1} = 0.6 \times 1 + 1 \times 1 = 1.6$$

The first encoded protein sequence present in Table 7.1 is composed of only three exchange groups e_2, e_4, and e_5 and the remaining exchange groups, i.e., e_1, e_3, and e_6 are absent in the first sequence (SEQ_1). Due to the absence of exchange groups e_1, e_3, and e_6 in SEQ_1, the positional weight ($(PW)_{ab}^{SEQ_o}$) corresponding to these exchange groups is zero. On the contrary, the exchange group e_2 occurs three times in the SEQ_1 at position 2, 4, and 8. Therefore, the

positional weight assign to the exchange group e_2 for each position is 1 which is multiplied by the probability of occurrence of exchange group e_2 on these positions, i.e., 1, 1, and 0.6 reported in Table 7.2.

Thus, the final weight of exchange group e_2 is determined by adding the product of positional weight and probability of occurrence of exchange group e_2 based on the presence in SEQ_1. Similarly, the weight of exchange groups e_4 and e_5 are computed. Finally, the feature vector for SEQ_1 is obtained as $\{(e_1, 0), (e_2, 2.6), (e_3, 0), (e_4, 1.4), (e_5, 1.6), (e_6, 0)\}$. Similarly, by using three stages of the CPSF approach, the feature vectors of all the protein sequences present in Figure 7.1 are determined and reported in Table 7.3. The complete flowchart showing the working of each stage of CPSF approach along with the classification of protein sequences into superfamilies is presented in Figure 7.2. It can be inferred that feature vectors generated by CPSF approach cover all the possible variations of amino acids by considering both the local and global similarity measures. In addition to this, the proposed CPSF approach represents a long chain of the protein sequence with a feature vector consists of only six dimensions (attributes). The obtained feature vectors with the proposed CPSF approach are used in several well-known classifiers for the efficient classification of protein sequences into superfamilies.

7.2 Experimental Evaluation

In this section, we present the details of the benchmark protein database comprises of four superfamilies, which are used to evaluate the performance of the proposed CPSF approach on standard classifiers such as SVM [170], NB [171], k-NN [172], and BLTA [173] classifiers. We have used WEKA [174] for implementing SVM, NB, and k-NN classifiers. To make it simple and easier, we just

Table 7.3: Representation of feature vector.

Sequence	e_1	e_2	e_3	e_4	e_5	e_6
1	0	2.6	0	1.4	1.6	0
2	0	3.2	0	0.6	1.6	0
3	0.4	2.4	0	0.4	1.4	0.2
4	0	2.8	0	0.6	2.0	0
5	0.4	2.6	0	1.6	1	0

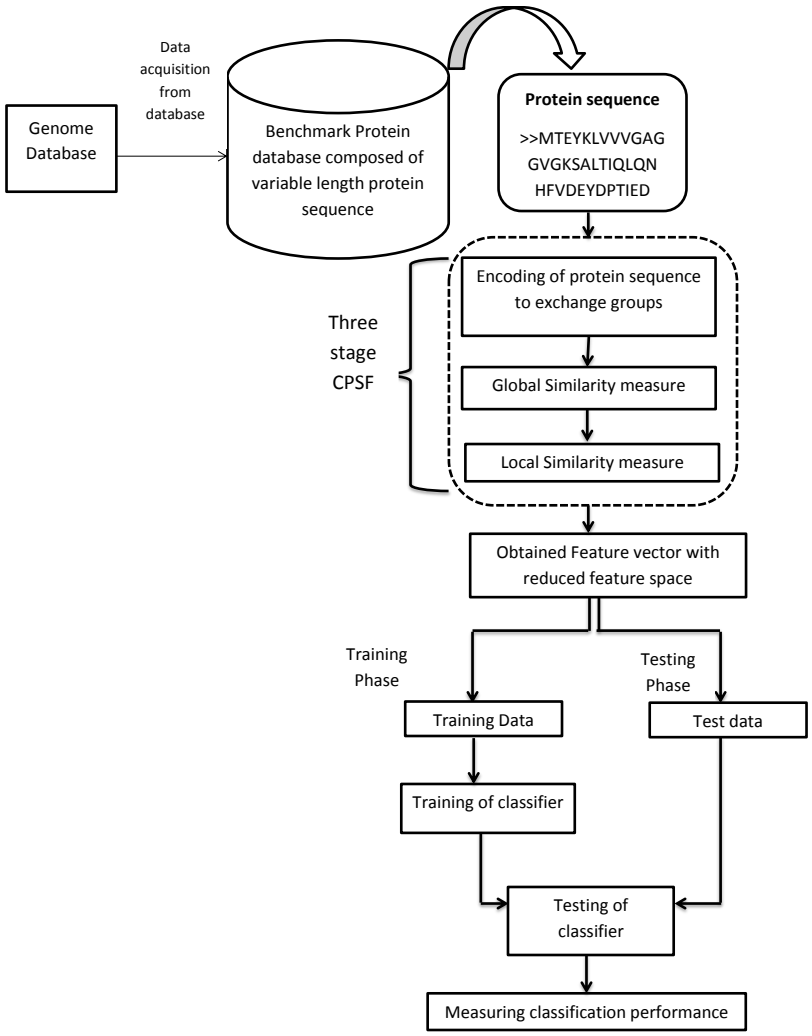


Figure 7.2: The flowchart of proposed CPSF approach.

use the default parameter values of these classifiers as found in WEKA [174]. In addition to these standard classifiers, we have implemented a neural network classifier called BLTA. For this classifier, we set its one of the parameters m-circle to 8. This m-circle refers to as the groups consisting of powers of 2 (which we will refer to as 2-circles, 4-circles, 8-circles and so on) is used as the basis of reduction in the Karnaugh domain [175]. The performance of the proposed CPSF approach is investigated on these classifiers in terms of various measures used for multi-class classification problems. The measures are confusion matrix, sensitivity, specificity, and classification accuracy [151]. The detailed description of these measures is already presented in the section 2.8.2. Furthermore, the performance of the CPSF approach is investigated in comparison with other feature extraction methods on these classifiers in terms of classification accuracy. Also, this section presents the details of the real-life database (a protein database of complex plant genome) of size 80GB collected from the DSR. The

preprocessing of this database is done using the CPSF approach, and then the extracted features are applied as an input to the SRSIO-FCM algorithm so that the protein sequences are efficiently assigned to the respective superfamilies. The performance of the SRSIO-FCM algorithm for this database is evaluated in terms of NMI [46], F-measure [45], and classification accuracy [151], respectively. The brief description of these measures is already presented in the section 2.8.1 and 2.8.2. The information about the protein databases and experimental settings used in the experimental evaluation are discussed next.

7.2.1 Datasets and Experimental Settings

To demonstrate the efficacy of the proposed CPSF approach on well-known classifiers, we have created a benchmark protein database composed of variable length protein sequences of four superfamilies, i.e., Ras, Globin, Trypsin, and Kinase collected from a publicly available database, i.e., UniProtKB [176]. The UniProtKB is a central repository to access complete protein sequences with functional information about various organisms and species. The benchmark protein database used here is composed of a different number of sequences taken from each superfamily. The information of the number of protein sequences taken from each superfamily is presented in Table 7.4, and the detailed description of these superfamilies is given in Appendix C. To evaluate the generalizability of our CPSF approach in comparison with other feature extraction approaches on well-known classifiers, we divided the training and testing data using 10-fold cross validation scheme. Also, the results are reported by dividing the training and testing data into 50-50, 60-40, and 70-30 training-testing partitions to show the importance of proper training. The detailed description of 10-fold cross validation scheme and different training-testing partitions are

Table 7.4: Information of the superfamilies of the benchmark protein database.

Name of superfamilies	Number of sequences	Minimum length of sequence	Maximum length of sequence
Ras	2460	171	296
Globin	2500	128	339
Trypsin	2510	142	485
Kinase	2470	215	860
Total sequences	9940	-	-

given in Appendix B. The classifiers with their default settings have been used in the experimental evaluation.

All the experiments are carried out on Intel(R) Xeon(R) E5-1607 v3 @ 3.10GHz x 4, workstation PC with 64 GB of memory and running on the Windows 7 Professional operating system. The implementation is done in the MATLAB computing environment and executed on MATLAB version R2014a. Furthermore, to investigate the efficacy of proposed CPSF on the SRSIO-FCM algorithm to handle the real-life genome data, the protein database is collected from the DSR, Indore which is of size 80 GB and consists of protein sequences of complex plant genomes. The detailed description of this protein database is given in Appendix D. All the experiments related to this protein database of size 80 GB are carried on the Apache Spark Cluster. The experiments have been done on the Spark Cluster with 3 nodes. Each node has the following configuration: Intel(R) Xenon(R) CPU E5-1607 v3 @ 3.10GHz x 4, 32GB RAM and 2TB storage.

7.2.2 Experimental Results and Discussion

In this section, we present the experimental results in terms of confusion matrix, sensitivity, specificity, and classification accuracy to evaluate the performance of the proposed CPSF approach on well-known classifiers, i.e., SVM [170], NB [171], k-NN [172], and BLTA [173] for solving the multi-class protein sequence classification problem. In addition to this, the performance of the CPSF approach compared with other feature extraction methods which are applied on the same standard classifiers. Furthermore, the proposed CPSF approach along with the SRSIO-FCM algorithm is applied to real-life complex plant genome data for its efficient categorization. The efficiency is reported in terms of NMI, F-measure, and classification accuracy, respectively.

Comparison of classification accuracy of CPSF approach on SVM, NB, k-NN, and BLTA classifiers for classification of protein sequences

The results of classification accuracy of proposed CPSF approach on SVM, NB, k-NN, and BLTA classifiers are presented in Table 7.5 using 50-50, 60-40, and 70-30 training-testing partitions and 10-fold cross validation scheme. It can be

seen from the results that the features extracted by the proposed CPSF approach when applied on SVM, NB, k-NN, and BLTA classifiers, the highest classification accuracy is achieved with SVM classifier for 50-50 and 60-40 training-testing partitions are 98.28% and 98.56%, respectively. This shows that even for smaller training samples, the features extracted by CPSF approach enhances the classification capability of SVM classifier and efficiently classify protein sequences into superfamilies. On the other hand, for 70-30 training-testing partition, the highest classification accuracy is achieved with NB, and k-NN classifiers are 100% which is outstandingly high. The higher classification accuracy shows the importance of proper training data provided as the features extracted by the CPSF approach for the efficient classification of protein sequences into superfamilies. For 70-30 training-testing partition, SVM and BLTA classifiers also show improvement in classification accuracy when the training data increases, but it is not equivalent to NB and k-NN classifiers. The classification accuracy of 10-fold cross validation scheme shows that SVM, NB, and k-NN classifiers achieves 100% classification, and classify protein sequences into superfamilies with high accuracy which is superior in comparison to BLTA classifier. Thus, it is inferred from the results that feature extracted by the proposed CPSF approach are highly efficient and statistically significant that leads to the improvement in classification accuracy of these classifiers for different training and testing partitions.

Table 7.5: Comparison of classification accuracies of proposed CPSF approach on several classifiers for classification of the benchmark protein database.

Validation methods	SVM Acc(%)			NB Acc(%)			k-NN Acc(%)			BLTA Acc(%)		
	Mean	SD	Max	Mean	SD	Max	Mean	SD	Max	Mean	SD	Max
50-50	97.48	0.249	98.28	96.28	0.250	97.09	95.13	0.521	96.21	96.02	0.138	96.46
60-40	98.04	0.124	98.56	97.49	0.874	98.08	96.27	0.451	97.73	97.12	0.898	97.91
70-30	98.82	0.483	99.98	99.10	0.295	100	98.93	0.477	100	97.89	0.506	98.82
10-fold cross validation	99.11	0.346	100	99.23	0.321	100	99.12	0.342	100	98.98	0.440	99.94

Comparison of sensitivity and specificity of CPSF approach on SVM, NB, k-NN, and BLTA classifiers for classification of benchmark protein sequences

The results of sensitivity and specificity in terms of mean and SD values of proposed CPSF approach on SVM, NB, k-NN, and BLTA classifiers are presented in Table 7.6 using 50-50, 60-40, and 70-30 training-testing partitions and 10-fold cross validation scheme. The reported results show that sensitivity and specificity achieved by the proposed CPSF approach on traditional classifiers for classification of protein sequences into superfamilies are significantly improved with the increase of training data. It can be inferred from the results that sensitivity and specificity of all the classifiers are highly appreciable with different validation methods for classification of protein sequences into superfamilies.

Comparison of actual and predicted classification of protein sequences by the proposed CPSF approach with well-known classifiers

The results are presented in the form of confusion matrix to show the actual and predicted classification of protein sequences performed by the proposed CPSF approach with SVM, NB, k-NN, and BLTA classifiers for each superfamily. The results are reported with different validation methods as reported in Table 7.7. The reported results indicate that the rate of true positive and true negative increases with the increase of training data. Especially, for 70-30 training-testing partition and 10-fold cross validation, the rate of true positive and the true negative sample is 100% in the case of most of the classifiers. The results show that the proposed approach extracts the statistically significant features when applied on well-known classifiers and gives a good generalization and adaptation capability of classifiers for classification of protein sequences into superfamilies.

Table 7.6: Comparison of sensitivity and specificity of proposed CPSF approach on several classifiers for classification of the benchmark protein database.

Validation methods	SVM				NB				k-NN				BLTA			
	Sensitivity		Specificity		Sensitivity		Specificity		Sensitivity		Specificity		Sensitivity		Specificity	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
50-50	96.52	0.256	98.04	0.347	93.82	0.253	97.01	0.443	91.63	0.752	96.19	0.528	92.04	1.266	97.23	0.426
60-40	96.72	0.356	98.78	0.526	93.36	0.324	97.28	0.652	93.85	0.652	96.85	0.652	96.15	1.796	97.16	1.598
70-30	98.13	0.642	99.01	0.621	98.89	1.054	99.17	0.452	98.98	1.086	99.31	0.254	96.04	1.873	98.01	0.624
10-fold cross validation	98.78	1.802	99.23	0.701	98.95	1.001	100	0	99.09	0.487	100	0	98.99	1.854	99.32	0.951

Table 7.7: Confusion matrix of proposed CPSF approach with well-known classifiers on the benchmark protein database with different validation methods.

Validation methods	Classifiers	Desired results	Output results for Ras-Globin-Trypsin-Kinase classification					Validation methods	Output results for Ras-Globin-Trypsin-Kinase classification				
			Ras	Globin	Trypsin	Kinase	% Score		Ras	Globin	Trypsin	Kinase	% Score
50-50	SVM	Ras	1193	15	12	10	96.99	60-40	966	4	5	9	98.17
		Globin	25	1203	12	10	96.24		9	975	7	9	97.5
		Trypsin	16	15	1212	12	96.57		6	16	963	19	95.91
		Kinase	10	23	10	1192	96.51		18	8	4	958	96.96
	NB	Ras	1170	25	10	25	95.12		954	20	6	4	96.95
		Globin	39	1179	11	21	94.32		30	958	8	4	95.8
		Trypsin	21	19	1181	34	94.1		7	16	962	19	95.81
		Kinase	40	18	26	1151	93.19		12	14	12	950	96.15
	k-NN	Ras	1143	52	18	17	92.92		947	10	10	17	96.23
		Globin	66	1150	22	12	92		15	948	17	20	94.8
		Trypsin	45	31	1161	18	92.5		13	15	956	20	95.21
		Kinase	50	30	15	1140	92.3		14	12	17	945	95.64
	BLTA	Ras	878	0	352	0	71.38		846	0	138	0	85.97
		Globin	0	1250	0	0	100		0	994	6	0	99.4
		Trypsin	0	0	1255	0	100		7	7	990	0	98.6
		Kinase	0	0	0	1235	100		4	0	4	980	99.19
70-30	SVM	Ras	737	1	0	0	99.86	10-fold	246	0	0	0	100
		Globin	0	750	0	0	100	cross	0	250	0	0	100
		Trypsin	0	0	753	0	100	validation	0	0	251	0	100
		Kinase	0	0	0	741	100		0	0	0	247	100
	NB	Ras	738	0	0	0	100		246	0	0	0	100
		Globin	0	750	0	0	100		0	250	0	0	100
		Trypsin	0	0	753	0	100		0	0	251	0	100
		Kinase	0	0	0	741	100		0	0	0	247	100
	k-NN	Ras	738	0	0	0	100		246	0	0	0	100
		Globin	0	750	0	0	100		0	250	0	0	100
		Trypsin	0	0	753	0	100		0	0	251	0	100
		Kinase	0	0	0	741	100		0	0	0	247	100
	BLTA	Ras	668	0	70	0	90.51		245	0	1	0	99.59
		Globin	0	750	0	0	100		0	250	0	0	100
		Trypsin	0	0	753	0	100		0	0	251	0	100
		Kinase	0	0	0	741	100		0	0	0	247	100

Comparison of classification accuracy of proposed CPSF approach with other feature extraction approaches on well-known classifiers

The performance of the proposed CPSF approach is evaluated in comparison with other feature extraction approaches discussed in the literature for classification of protein sequence into superfamilies. The feature encoding technique used along with the number of features extracted by each approach is listed here in Table 7.8. We have evaluated the performance of the CPSF approach in comparison with other feature extraction approaches in terms of classification accuracy and reported results in terms of mean, SD, and maximum accuracies for 50-50, 60-40, and 70-30 training-testing partitions and 10-fold cross validation scheme, respectively in Table 7.9.

The reported results show that the classification accuracies achieved by

Table 7.8: Comparison of the number of features extracted with different feature extraction approaches on the benchmark protein database.

Authors	Feature extraction method	Number of features
Bandyopadhyay [144]	l-gram feature encoding method	20
Mansoori et al. [145]	Exchange group encoding method	6
Iqbal et al. [149]	Sequence encoding and statistical measure based feature selection method	20
This study	CPSF method	6

the proposed CPSF approach on different classifiers are comparatively much higher than the classification accuracies achieved by other features extraction approaches on these classifiers. This shows that although the number of features extracted by the proposed CPSF approach is equal to the number of features extracted by mansoori et al. [145] and much lesser than the features extracted by Bandyopadhyay [144] and Iqbal et al. [149], but the classification accuracy achieved by the CPSF approach is significantly higher. This is because the proposed CPSF approach captures both the statistical characteristics along with positional information of amino acids in protein sequences more efficiently. Thus, the reported results affirm the potential use of the proposed method for more accurate categorization of protein sequences into superfamilies.

Table 7.9: Comparison of classification accuracies of feature extraction approaches on well-known classifiers for the benchmark protein database.

Classifiers	Validation methods	Bandyopadhyay [144] Acc(%)			Mansoori et al. [145] Acc(%)			Iqbal et al. [149] Acc(%)			CPSF approach Acc(%)		
		Mean	SD	Max	Mean	SD	Max	Mean	SD	Max	Mean	SD	Max
SVM	50-50	87.90	0.671	88.21	74.90	0.754	75.19	74.48	1.09	75.67	97.48	0.249	98.28
	60-40	88.09	0.912	89.11	75.18	0.597	76.33	75.01	0.659	76.11	98.04	0.124	98.56
	70-30	88.85	0.789	89.10	77.09	0.367	77.18	78.81	0.359	79.67	98.82	0.483	99.98
	10-fold	89.97	0.768	90.57	79.55	0.991	80.08	79.33	0.559	80.85	99.11	0.346	100
NB	50-50	85.49	0.651	86.14	70.95	0.897	71.42	78.97	1.78	80.99	96.28	0.25	97.09
	60-40	86.11	0.776	86.91	71.89	0.568	72.21	79.31	0.989	81.87	97.49	0.874	98.08
	70-30	86.99	0.589	87.03	73.45	0.923	74.19	81.19	0.886	82.11	99.10	0.295	100
	10-fold	87.78	0.91	88.91	75.88	0.398	76.61	83.35	0.328	84.12	99.23	0.321	100
k-NN	50-50	84.88	0.897	85.73	74.18	1.25	75.93	83.99	0.119	84.17	95.13	0.521	96.21
	60-40	84.97	0.988	85.99	75.33	0.449	75.88	85.48	0.449	86.88	96.27	0.451	97.73
	70-30	85.89	0.987	86.19	76.61	0.783	77.11	87.11	0.878	88.92	98.93	0.477	100
	10-fold	86.10	0.897	86.94	79.10	1.11	81.93	89.66	0.187	90.28	99.12	0.342	100
BLTA	50-50	82.59	0.875	83.08	73.01	0.165	73.89	92.96	0.839	93.69	96.02	0.138	96.46
	60-40	84.01	0.901	85.89	73.98	0.697	74.67	93.71	0.671	94.19	97.12	0.898	97.91
	70-30	85.55	0.152	86.13	75.99	0.122	76.83	94.5	0.991	95.89	97.89	0.506	98.82
	10-fold	87.98	0.187	88.61	78.12	0.089	78.65	96.69	0.39	97.01	98.98	0.44	99.94

Performance evaluation of real-life Big Data collected from DSR

As discussed earlier, the proposed CPSF approach on well-known classifiers is validated on a benchmark protein database. Then, the proposed CPSF approach is applied to a real-life Big Data of size 80 GB (i.e., a protein database composed of various superfamilies of the complex plant genome) which is collected from the DSR, Indore. This huge protein database is preprocessed efficiently using the proposed CPSF feature extraction approach. This approach represents all the protein sequences present in this database in terms of fixed-length numeric feature vectors, where each feature vector consists of six dimensions. Then, the preprocessed database is passed as an input to the SRSIO-FCM algorithm for the efficient categorization of the genome database into various superfamilies. For this, the SRSIO-FCM algorithm firstly partitioned the entire protein database into various clusters by considering similarity among the protein sequences. Then, these formed clusters are used further to predict the superfamilies of unknown protein sequences to enhance the productivity of next generation protein sequences of a plant.

The performance of the SRSIO-FCM algorithm on the features extracted with the CPSF approach is evaluated on this massive protein database in terms of NMI, F-measure and classification accuracy, respectively and the results are given in Table 7.10. The higher value of NMI, F-measure, and classification accuracy indicates good categorization results. Thus, the reported results show that the CPSF approach with the SRSIO-FCM algorithm achieves remarkable results on this real-life protein database. The remarkable results help to assist the DSR Scientists in enhancing the productivity of next generation protein sequences of a plant.

Table 7.10: Result of the proposed approaches on real-life protein database.

Superfamilies	Measures		
	NMI	F-measure	Classification Accuracy (%)
ABB36645.1	0.7831	0.8657	91.85
ACI31552.1	0.7765	0.8563	87.98
AEE29332.1	0.7231	0.7891	83.45
AF456323.1	0.8897	0.9256	95.78
AGO06072.1	0.9123	0.9511	97.18
AHF074N12.027	0.8623	0.9141	94.86
AHF417E07.003	0.8232	0.8871	94.78
Arabidopsis_thaliana	0.9332	0.9902	99.22
BAE02726.1	0.8129	0.8762	93.98
gm_Glyma12g34160	0.7653	0.8231	84.67
Peanut	0.8097	0.8736	92.97
Plant_db_allgery_2	0.9156	0.9678	98.81
Plant_db_disease	0.8227	0.8859	94.06
Plant_db_sweet_gene	0.9106	0.9347	96.09
Plant_db_unknown	0.9256	0.9876	99.01
Plant_db_allergy	0.7651	0.8001	84.41
XP_003544375.1	0.9501	0.9941	99.89

7.3 Summary

In this chapter, the proposed SRSIO-FCM algorithm is applied to a real-life genome database collected from the DSR, Indore. For the preprocessing of this genome database, a novel CPSF approach is designed. First of all, the genome database gathered from the DSR, Indore is analyzed. Then, this genome database consists of a huge volume of variable length protein sequences is preprocessed using the proposed CPSF approach. This feature extraction approach works in three stages, i.e., in the first stage, it performs encoding of protein sequences where each protein sequence is represented in terms of exchange groups. Then, in the second and third stage, it computes the global similarity measure and local similarity measure for representing a protein sequence with a fixed-length numeric feature vector. One distinctive characteristic of the proposed CPSF approach is that it computes both local and global similarity for taking into account all possible position-specific variations of amino acids in a protein sequence as well as within a superfamily. The other important characteristic is that it represents each variable length protein sequence consist of a long chain

of amino acids with a fixed-length numeric vector consist of only six dimensions.

To validate the proposed CPSF approach, it is tested on a benchmark protein database comprises of four superfamilies and its performance is evaluated on SVM, NB, k-NN, and BLTA classifiers in terms of various performance measures such as confusion matrix, sensitivity, specificity, and classification accuracy, respectively. The protein sequences belong to four superfamilies are relatively long and have a very low sequence similarity among them. Therefore, it is very tough to classify protein sequences into existing superfamilies with high accuracy. The results show that features extracted by the proposed approach are highly useful and statistically significant. Thus, the features extracted with the proposed approach lead to a substantial improvement in the classification accuracy, sensitivity, and specificity of these classifiers for different training and testing partitions. Also, the results show that the rate of true positive and true negative sequences increases with the increase of training data this gives a good generalization and adaptation capability of the proposed CPSF approach in comparison with other state-of-the-art methods on these classifiers.

Furthermore, the proposed CPSF approach is applied to a real-life protein database collected from the DSR, Indore. The CPSF approach represents all protein sequences present in this database in terms of fixed-length numeric feature vectors. This preprocessed database is passed as an input to the SRSIO-FCM algorithm for its efficient categorization to the respective superfamilies. The experimental results on this massive protein database are reported in terms of NMI, F-measure, and classification accuracy, respectively. The higher value of NMI, F-measure, and classification accuracy indicates good categorization results. This shows that integration of proposed feature extraction approach with the SRSIO-FCM algorithm performs efficient categorization of protein sequences of complex plant genome into superfamilies.

Chapter 8

Conclusion

8.1 Contributions

Clustering is one of the most well-known concepts for discovering the underlying structure of data by assigning the data points into groups called clusters so that data points belong to the same group are more similar than the data points lying in different groups. The challenges in real-life situations are that every dataset has different geometrical distribution in feature space and there exists an ambiguity in designating a data point to a particular group or cluster. Therefore, it is very difficult to capture the underlying structure of different datasets with a single clustering algorithm. To handle such challenges, fuzzy clustering has emerged in recent years that solves various categorization problems. Thus, there is an immense need to analyze this kind of clustering to improve overall clustering processes which can handle different categories of data, such as Large, VL, and Big Data based on their size.

To achieve enhanced fuzzy clustering performance, we have designed a novel cluster validity index by combining three proposed measures named as Intra-cluster Compactness, Inter-cluster Separation based on Fuzzy sets, and Inter-cluster Overlap. The purpose of this work is to find the optimal number of clusters and the corresponding fuzzy partitions for the validation of fuzzy clustering. One of the distinctive characteristics of this validity index is that it minimizes the maximum dispersion so as to increase the overall compactness within the clusters. Also, the validity index maximizes the distance between the centers of less separated fuzzy clusters to increase the overall separation among all pairs of

fuzzy clusters and also minimizes the degree of overlap among the highly overlapped fuzzy clusters in order to reduce the overall overlapping among the fuzzy partitions. Hence, the purpose of a designed novel cluster validity index is to determine the optimal number of clusters such that the obtained clusters are expected to have high compactness, less degree of overlap, and higher separation. The designed cluster validity index outperforms the other compared indexes in terms of identifying the optimal number of clusters and the corresponding fuzzy partitions. In addition to this, the optimal number of clusters determined by the proposed cluster validity index is insensitive to a change in values of the fuzzifier and therefore it is considered as the most reliable cluster validity index.

Furthermore, for a better selection of other parameters of clustering such as the fuzzifier, the number of clusters and the locations of cluster centers, we have designed two hybrid fuzzy clustering algorithms named as the QIE-FCM and the EQIE-FCM algorithms. In the QIE-FCM algorithm, the quantum computing principle is utilized in fuzzy clustering to find the best value of the fuzzifier from a large search space after several generations of evolution by maintaining a proper balance between exploration and exploitation. In this approach, corresponding to the best value of fuzzifier, it determines the optimal number of clusters. One of the major limitations of the QIE-FCM approach is a random initialization of locations of cluster centers. Due to this, it takes a higher number of iterations to determine the proper locations of cluster centers, and thus it leads to a slow convergence problem. To overcome this problem, the EQIE-FCM algorithm is designed that uses the quantum computing principle in fuzzy clustering to find the optimal parameters of fuzzy clustering, i.e., the fuzzifier, the number of clusters, and the location of cluster centroids from a large search space after several generations of evolution by keeping a proper balance between exploration and exploitation. The performance of the QIE-FCM algorithm has been evaluated in comparison with well-known cluster validity indexes and has also been compared to several evolutionary algorithms. It is found that the QIE-FCM algorithm achieves the minimum value of fitness function, identifies the best value of fuzzifier, and the optimal number of clusters. The performance of the EQIE-FCM algorithm is compared with the QIE-FCM algorithm and with other evolutionary clustering algorithms. From our experimental result, we can conclude that the EQIE-FCM algorithm is very efficient in terms of proper se-

lection of learning parameters, and hence, it converges to the optimal value of parameters and also achieves fast convergence as compared to the QIE-FCM algorithm.

Looking towards the current need of the clustering process to handle different categories of data, such as Large, VL and Big Data in real-life situations, another contribution is made to design a different fuzzy clustering process to handle VL data, and further scaled this to handle Big Data. For this, an incremental clustering algorithm called RSIO-FCM is designed to handle VL data. This processes data, chunk by chunk and performs clustering on each chunk by feeding forward final cluster centers of the present chunk for clustering of the next chunk. The main contribution made to the literature over other existing one is that this approach resolves the problem of overlapping locations of cluster centers. Furthermore, this approach is modeled as a classifier by using the Bayes' theorem that relates clustering results to the classification process for handling the classification of multi-class VL data. The reported results show that the RSIO-FCM efficiently handled VL data by showing improvement in classification accuracy with a significant reduction in run-time. One of the major limitation of the RSIO-FCM algorithm is that it produces highly deviated cluster centers in the case when two chunks would consist of samples belonging to distinct classes. Due to this reason, if the cluster centers of the previous subset are fed as an input for the clustering of the current subset, then it will converge slowly by taking a higher number of iterations during clustering of the current subset. Therefore, another contribution is made to overcome this problem and also capable of handling Big Data.

We contributed to the design and implementation of a novel SRSIO-FCM algorithm. This algorithm partitioned the Big Data into various chunks, and processed the data points present within the chunk in a parallel manner. One distinctive characteristic of the SRSIO-FCM is that due to the parallel processing of data points within the chunk, it achieves a significant reduction in run-time for the clustering of such huge amounts of data, without compromising the quality of clustering. The other important characteristic is that during the execution of the proposed algorithm, we eliminate the need for storing the large membership matrix, which significantly reduces the run-time and storage space and thus, it makes the execution of the proposed algorithm much faster. The

SRSIO-FCM algorithm is implemented using the Big Data processing framework called Apache Spark to deal with the challenges associated with fuzzy clustering for handling Big Data. For comparing the performance of SRSIO-FCM, two basic algorithms present in literature are enhanced by making these algorithms scalable for handling Big Data. The detailed experimentation is carried out and the reported results in the previous chapters, show the dominance of the proposed SRSIO-FCM algorithm over other comparable approaches in terms of various measures assessing the quality of clustering. The results show that the SRSIO-FCM has a great potential in Big Data clustering.

The proposed SRSIO-FCM algorithm is applied to a real-life Big Data classification problem (i.e., protein sequences of complex plant genome) for enhancing the productivity of next generation protein sequences of various species of plant. The productivity of next generation protein sequences can be enhanced by making protein sequences resistance to disease, drought, heat, high temperature, and food allergies. For this, a real-life Big Data is collected from the DSR, Indore which is of size 80 GB. First of all, this protein database is required to be preprocessed efficiently for its proper clustering and classification. For this reason, we have proposed a novel CPSF approach for feature extraction. One distinctive characteristic of the CPSF approach is that it computes both local and global similarity for taking into account all possible position-specific variations of amino acids in a protein sequence as well as within a superfamily. The other important characteristic is that it represents each variable length protein sequence consisting of a long chain of amino acids with a fixed-length numeric vector consist of only six dimensions. The exhaustive experiments reported in the previous chapters show the efficacy of the proposed CPSF approach on well-known classifier in terms of classification of protein sequences into superfamilies. Furthermore, the proposed CPSF approach is applied to the real-life protein database for its preprocessing. This preprocessed database is passed as an input to the SRSIO-FCM algorithm for efficient categorization of protein sequences of complex plant genome into superfamilies. The exhaustive results on this massive protein database are evaluated in terms of various performance measures. The reported results show that integration of CPSF approach with SRSIO-FCM algorithm performs efficient categorization of protein sequences into superfamilies to enhance the productivity of next generation protein se-

quences of various species of plant.

8.2 Future Work

There are some possible extensions of proposed approaches presented in the previous chapters. One of the most visible extensions of the problem is to assess the quality of VL and Big Data clustering. Thus, one of such measures, VI_{DSO} index, which is proposed herein needs to be redesigned to handle VL data and Big Data [36, 177]. Furthermore, fuzzy clustering is mainly limited to spherical clusters. To address this problem, kernel functions based fuzzy clustering can be designed by mapping data with nonlinear relationships to appropriate feature spaces for efficient kernel clustering of a different shape of datasets [178]. Thus, another possible extension is to design fuzzy clustering algorithms by making use of kernel functions. Further, kernel functions based scalable fuzzy clustering algorithms can also be developed to achieve better efficiency [36]. Investigation of the behavior of proposed scalable clustering algorithms on other genome databases like DNA and SNP is also an attractive solution for the DSR scientists.

Bibliography

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[2] V. Estivill Castro, "Why so many clustering algorithms: a position paper," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 1, pp. 65–75, 2002.

[3] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, "Automated variable weighting in k-means type clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 657–668, 2005.

[4] W. Cai, S. Chen, and D. Zhang, "A multiobjective simultaneous learning framework for clustering and classification," *IEEE transactions on neural networks*, vol. 21, no. 2, pp. 185–200, 2010.

[5] C. H. Wu, C. S. Ouyang, L. W. Chen, and L. W. Lu, "A new fuzzy clustering validity index with a median factor for centroid-based clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 3, pp. 701–718, 2015.

[6] C. Hennig, "What are the true clusters?" *Pattern Recognition Letters*, vol. 64, pp. 53–62, 2015.

[7] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[8] N. A. Erilli, U. Yolcu, E. Egrioglu, C. Aladag, and Y. oner, "Determining the most proper number of cluster in fuzzy clustering by using artificial neural networks," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2248–2252, 2011.

[9] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[10] G. Castellano, A. M. Fanelli, and C. Mencar, "A fuzzy clustering approach for mining diagnostic rules," in *Proc. of 2003 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, Washington, D.C., USA, October, 2003, pp. 2007–2012.

[11] H. B. Shen, J. Yang, X. J. Liu, and K. C. Chou, "Using supervised fuzzy clustering to predict protein structural classes," *Biochemical and Biophysical Research Communications*, vol. 334, no. 2, pp. 577–581, 2005.

[12] L. Jing, M. K. Ng, and J. Z. Huang, "An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1026–1041, 2007.

[13] Z. Deng, K. S. Choi, F. L. Chung, and S. Wang, "Enhanced soft subspace clustering integrating within-cluster and between-cluster information," *Pattern Recognition*, vol. 43, no. 3, pp. 767–781, 2010.

[14] M. Sardana and R. Agrawal, "A comparative study of clustering methods for relevant gene selection in microarray data," in *Proc. of Second International Conference on Computer Science, Engineering and Applications, Advances in Computer Science, Engineering and Applications*, ser. Advances in Intelligent and Soft Computing. Springer Berlin Heidelberg, 2012, vol. 166, pp. 789–797.

[15] L. Tang, H. Liu, and J. Zhang, "Identifying evolving groups in dynamic multimode networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 1, pp. 72–85, 2012.

[16] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[17] Y. Leung, J. S. Zhang, and Z. B. Xu, "Clustering by scale-space filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1396–1410, 2000.

[18] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis*. Wiley Interscience New York, 1973, vol. 3.

[19] R. Bellman, R. Kalaba, and L. Zadeh, "Abstraction and pattern classification," *Journal of Mathematical Analysis and Applications*, vol. 13, no. 1, pp. 1–7, 1966.

[20] K. Zhou, C. Fu, and S. Yang, "Fuzziness parameter selection in fuzzy c-means: the perspective of cluster validation," *Science China Information Sciences*, vol. 57, no. 11, pp. 1–8, 2014.

[21] A. T. Azar, S. A. El-Said, and A. E. Hassanien, "Fuzzy and hard clustering analysis for thyroid disease," *Computer methods and programs in biomedicine*, vol. 111, no. 1, pp. 1–16, 2013.

[22] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.

[23] S. Ghosh, S. Biswas, D. Sarkar, and P. P. Sarkar, "A novel neuro-fuzzy classification technique for data mining," *Egyptian Informatics Journal*, vol. 15, no. 3, pp. 129–147, 2014.

[24] M. Huang, Z. Xia, H. Wang, Q. Zeng, and Q. Wang, "The range of the value for the fuzzifier of the fuzzy c-means algorithm," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2280–2284, 2012.

[25] K. L. Wu, "Analysis of parameter selections for fuzzy c-means," *Pattern Recognition*, vol. 45, no. 1, pp. 407–415, 2012.

[26] M. Erisoglu, N. Calis, and S. Sakallioglu, "A new algorithm for initial cluster centers in k-means algorithm," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1701–1705, 2011.

[27] H. Izakian and A. Abraham, "Fuzzy c-means and fuzzy swarm for fuzzy clustering problem," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1835–1838, 2011.

[28] P. J. Huber, "Massive data sets workshop: the morning after," in *Proc. of a Workshop Massive Data Sets*. Washington, DC: National Academy Press, 1997, pp. 169–184.

[29] R. J. Hathaway and J. C. Bezdek, "Extending fuzzy and probabilistic clustering to very large data sets," *Computational Statistics and Data Analysis*, vol. 51, no. 1, pp. 215–234, 2006.

[30] T. W. Cheng, D. B. Goldgof, and L. O. Hall, "Fast fuzzy clustering," *Fuzzy Sets and Systems*, vol. 93, no. 1, pp. 49–56, 1998.

[31] M. C. Hung and D. L. Yang, "An efficient fuzzy c-means clustering algorithm," in *Proc. of 2001 IEEE International Conference on Data Mining*. IEEE, San Jose, California, USA, December, 2001, pp. 225–232.

[32] N. R. Pal and J. C. Bezdek, "Complexity reduction for large image processing," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 32, no. 5, pp. 598–611, 2002.

[33] P. Hore, L. O. Hall, and D. B. Goldgof, "Single pass fuzzy c means," in *Proc. of 2007 IEEE International on Fuzzy Systems Conference.* IEEE, London, United Kingdom, July, 2007, pp. 1–7.

[34] L. Wang, J. C. Bezdek, C. Leckie, and R. Kotagiri, "Selective sampling for approximate clustering of very large data sets," *International Journal of Intelligent Systems*, vol. 23, no. 3, pp. 313–331, 2008.

[35] P. Hore, L. O. Hall, D. B. Goldgof, Y. Gu, A. A. Maudsley, and A. Darkazanli, "A scalable framework for segmenting magnetic resonance images," *Journal of Signal Processing Systems*, vol. 54, no. 1-3, pp. 183–203, 2009.

[36] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 6, pp. 1130–1146, 2012.

[37] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms.* Springer Science and Business Media, 2013.

[38] P. Hore, L. Hall, D. Goldgof, and W. Cheng, "Online fuzzy c means," in *Proc. of 2008 Annual Meeting of the North American Fuzzy Information Processing Society.* IEEE, New York City, NY, USA, May, 2008, pp. 1–5.

[39] S. Eschrich, J. Ke, L. O. Hall, and D. B. Goldgof, "Fast accurate fuzzy clustering through data reduction," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 2, pp. 262–270, 2003.

[40] Y. Zhai, Y. S. Ong, and I. W. Tsang, "The emerging big dimensionality," *IEEE Computational Intelligence Magazine*, vol. 9, no. 3, pp. 14–26, 2014.

[41] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: a new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[42] S. Bandyopadhyay and U. Maulik, "Nonparametric genetic clustering: comparison of validity indices," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 1, pp. 120–125, 2001.

[43] C. C. Hung, E. Casper, B. C. Kuo, W. Liu, X. Yu, E. Jung, and M. Yang, "A quantum-modeled fuzzy c-means clustering algorithm for remotely sensed multi-band image segmentation," in *Proc. of 2013 IEEE International Geoscience and Remote Sensing Symposium.* IEEE, Melbourne, Australia, July, 2013, pp. 2501–2504.

[44] T. M. Silva Filho, B. A. Pimentel, R. M. Souza, and A. L. Oliveira, "Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization," *Expert Systems with Applications*, vol. 42, no. 17, pp. 6315–6328, 2015.

[45] Y. Wang, L. Chen, and J. P. Mei, "Incremental fuzzy clustering with multiple medoids for large data," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 6, pp. 1557–1568, 2014.

[46] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[47] K. Y. Yeung and W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.

[48] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *Proc. of 19th International Conference on Artificial neural networks, Artificial Neural Networks ICANN 2009*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5769, pp. 175–184.

[49] X. Xu, J. Jager, and H. P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," in *Proc of Scaling Algorithms, Applications and Systems, High Performance Data Mining*. Springer US, 1999, pp. 263–290.

[50] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Proc. of IEEE International Conference on Cloud Computing, Cloud Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5931, pp. 674–679.

[51] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[52] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[53] S. Khan, G. Situ, K. Decker, and C. J. Schmidt, "Gofigure: Automated gene ontology annotation," *Bioinformatics*, vol. 19, no. 18, pp. 2484–2485, 2003.

[54] S. Gunnemann, H. Kremer, D. Lenhard, and T. Seidl, "Subspace clustering for indexing high dimensional data: a main memory index based on local reductions and individual multi-representations," in *Proc. of 14th International Conference on Extending Database Technology*. ACM, Uppsala, Sweden, March, 2011, pp. 237–248.

[55] A. Ducournau, A. Bretto, S. Rital, and B. Laget, "A reductive approach to hypergraph clustering: an application to image segmentation," *Pattern Recognition*, vol. 45, no. 7, pp. 2788–2803, 2012.

[56] T. F. Ng, T. D. Pham, and X. Jia, "Feature interaction in subspace clustering using the choquet integral," *Pattern Recognition*, vol. 45, no. 7, pp. 2645–2660, 2012.

[57] J. Y. Jiang, R. J. Liou, and S. J. Lee, "A fuzzy self-constructing feature clustering algorithm for text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 3, pp. 335–349, 2011.

[58] J. Feyereisl and U. Aickelin, "Privileged information for data clustering," *Information Sciences*, vol. 194, pp. 4–23, 2012.

[59] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier, "Generalized k-means-based clustering for temporal data under weighted and kernel time warp," *Pattern Recognition Letters*, vol. 75, pp. 63–69, 2016.

[60] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[61] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, 2009, vol. 344.

[62] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[63] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. University of California Press, 1967, pp. 281–297.

[64] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009.

[65] C. D. Manning, P. Raghavan, H. Schutze *et al.*, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 1, no. 1.

[66] A. Kandel and W. J. Byatt, "Fuzzy sets, fuzzy algebra, and fuzzy statistics," *Proceedings of the IEEE*, vol. 66, no. 12, pp. 1619–1639, 1978.

[67] J. K. Parker, L. O. Hall, and A. Kandel, "Scalable fuzzy neighborhood dbscan," in *Proc. of 2010 IEEE International Conference on Fuzzy Systems*. IEEE, Barcelona, Spain, July, 2010, pp. 1–8.

[68] J. K. Parker, L. O. Hall, and J. C. Bezdek, "Comparison of scalable fuzzy clustering methods," in *Proc. of 2012 IEEE International Conference on Fuzzy Systems*. IEEE, Brisbane, Australia, June, 2012, pp. 1–9.

[69] F. D. A. De Carvalho and C. P. Tenorio, "Fuzzy k-means clustering algorithms for interval-valued data based on adaptive quadratic distances," *Fuzzy Sets and Systems*, vol. 161, no. 23, pp. 2978–2999, 2010.

[70] M. Sato-Ilic, "Symbolic clustering with interval-valued data," *Procedia Computer Science*, vol. 6, pp. 358–363, 2011.

[71] H. Zhao, Z. Xu, S. Liu, and Z. Wang, "Intuitionistic fuzzy mst clustering algorithms," *Computers and Industrial Engineering*, vol. 62, no. 4, pp. 1130–1140, 2012.

[72] C. W. De Almeida, R. M. De Souza, and A. L. Candeias, "Fuzzy kohonen clustering networks for interval data," *Neurocomputing*, vol. 99, pp. 65–75, 2013.

[73] J. C. Dunn, *A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters*. Taylor and Francis, 1973.

[74] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.

[75] J. Bezdek, *Pattern recognition in handbook of fuzzy computation*. Boston, NY, 1998.

[76] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 8, pp. 841–847, 1991.

[77] K. L. Wu and M. S. Yang, "Alternative c-means clustering algorithms," *Pattern Recognition*, vol. 35, no. 10, pp. 2267–2278, 2002.

[78] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-means clustering," *Pattern Recognition Letters*, vol. 25, no. 11, pp. 1293–1302, 2004.

[79] A. Bahrololoum, H. Nezamabadi-pour, and S. Saryazdi, "A data clustering approach based on universal gravity rule," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 415–428, 2015.

[80] D. Binu, "Cluster analysis using optimization algorithms with newly designed objective functions," *Expert Systems with Applications*, vol. 42, no. 14, pp. 5848–5859, 2015.

[81] J. K. Parker and L. O. Hall, "Accelerating fuzzy-c means using an estimated subsample size," *IEEE Transactions on Fuzzy Systems,*, vol. 22, no. 5, pp. 1229–1244, 2014.

[82] J. Yu, Q. Cheng, and H. Huang, "Analysis of the weighting exponent in the fcm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 634–639, 2004.

[83] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "Validity index for crisp and fuzzy clusters," *Pattern Recognition*, vol. 37, no. 3, pp. 487–501, 2004.

[84] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 746–774, 2001.

[85] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[86] K. R. Žalik and B. Žalik, "Validity index for clusters of different sizes and densities," *Pattern Recognition Letters*, vol. 32, no. 2, pp. 221–234, 2011.

[87] S. Ramanna, P. Lingras, C. Sombattheera, and A. Krishna, "Multi-disciplinary trends in artificial intelligence," in *Proc. of 7th International Workshop on Multi-disciplinary Trends in Artificial Intelligence -MIWAI 2013*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2013, vol. 8271.

[88] R. Xu, J. Xu, and D. C. Wunsch, "A comparison study of validity indices on swarm-intelligence-based clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 4, pp. 1243–1256, 2012.

[89] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," in *Proc. of 2001 IEEE International Conference on Data Mining.* IEEE, San Jose, California, USA, December, 2001, pp. 187–194.

[90] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2008.

[91] J. C. Bezdek, "Cluster validity with fuzzy sets," *Journal of Cybernatics*, vol. 3, no. 3, pp. 58–73, 1973.

[92] Y. Fukuyama and M. Sugeno, "A new method of choosing the number of clusters for the fuzzy c-means method," in *Proc. of 5th Symposium on Fuzzy Systems*, vol. 247, 1989, pp. 247–250.

[93] M. R. Rezaee, B. P. Lelieveldt, and J. H. Reiber, "A new cluster validity index for the fuzzy c-mean," *Pattern Recognition Letters*, vol. 19, no. 3, pp. 237–246, 1998.

[94] C. H. Chou, M. C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," *Pattern Analysis and Applications*, vol. 7, no. 2, pp. 205–220, 2004.

[95] D. W. Kim, K. H. Lee, and D. Lee, "On cluster validity index for estimation of the optimal number of fuzzy clusters," *Pattern Recognition*, vol. 37, no. 10, pp. 2009–2025, 2004.

[96] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification," *Fuzzy Sets and Systems*, vol. 155, no. 2, pp. 191–214, 2005.

[97] K. L. Wu and M. S. Yang, "A cluster validity index for fuzzy clustering," *Pattern Recognition Letters*, vol. 26, no. 9, pp. 1275–1291, 2005.

[98] H. L. Capitaine and C. Frelicot, "A cluster-validity index combining an overlap measure and a separation measure based on fuzzy-aggregation

operators," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 3, pp. 580–588, 2011.

[99] H. Lee Kwang, Y. S. Song, and K. M. Lee, "Similarity measure between fuzzy sets and between elements," *Fuzzy Sets and Systems*, vol. 62, no. 3, pp. 291–293, 1994.

[100] L. Mascarilla, M. Berthier, and C. Frelicot, "A k-order fuzzy or operator for pattern classification with k-order ambiguity rejection," *Fuzzy Sets and Systems*, vol. 159, no. 15, pp. 2011–2029, 2008.

[101] L. O. Hall, A. M. Bensaid, L. P. Clarke, R. P. Velthuizen, M. S. Silbiger, and J. C. Bezdek, "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 672–682, 1992.

[102] M. J. Fadili, S. Ruan, D. Bloyet, and B. Mazoyer, "On the number of clusters and the fuzziness index for unsupervised fca application to bold fmri time series," *Medical Image Analysis*, vol. 5, no. 1, pp. 55–67, 2001.

[103] S. Kannan, S. Ramathilagam, and P. Chung, "Effective fuzzy c-means clustering algorithms for data clustering problems," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6292–6300, 2012.

[104] H. Wang, W. Zhu, J. Liu, L. Li, and Z. Yin, "Multidistribution center location based on real-parameter quantum evolutionary clustering algorithm," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[105] S. Kannan, S. Ramathilagam, R. Devi, and A. Sathya, "Robust kernel fcm in segmentation of breast medical images," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4382–4389, 2011.

[106] G. Gan, J. Wu, and Z. Yang, "A genetic fuzzy k-modes algorithm for clustering categorical data," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1615–1620, 2009.

[107] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[108] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy, "Finding a better-than-classical quantum and/or algorithm using genetic programming," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999, pp. 2239–2246.

[109] K. H. Ĥan and J. H. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, h_ε gate, and two-phase scheme," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 156–169, 2004.

[110] T. C. Lu, G. R. Yu, and J. C. Juang, "Quantum-based algorithm for optimizing artificial neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 8, pp. 1266–1278, 2013.

[111] J. F. Kolen and T. Hutcheson, "Reducing the time complexity of the fuzzy c-means algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 263–267, 2002.

[112] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research," *ASU Feature Selection Repository*, pp. 1–28, 2010.

[113] Y. Zhai, M. Tan, I. Tsang, and Y. S. Ong, "Discovering support and affiliated features from very high dimensions," *arXiv preprint arXiv:1206.6477*, 2012.

[114] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[115] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[116] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*. ACM, 1998, vol. 27, no. 2.

[117] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[118] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proc. of Thirty-Sixth Annual ACM Symposium on Theory of Computing*. ACM, Chicago, IL, USA, June, 2004, pp. 291–300.

[119] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *Proc. of 1998 ACM SIGMOD International*

Conference on Management of Data, vol. 27, no. 2. ACM, Seattle, Washington, USA, June, 1998, pp. 73–84.

[120] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[121] P. Mika, "Flink: Semantic web technology for the extraction and analysis of social networks," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2, pp. 211–223, 2005.

[122] A. M. Team, "Apache mahout: Scalable machine-learning and data-mining library," 2011. [Online]. Available: http://mahout.apache.org/

[123] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: efficient iterative data processing on large clusters," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 285–296, 2010.

[124] J. Xu, Z. Chen, J. Tang, and S. Su, "T-storm: traffic-aware online scheduling in storm," in *Proc. of IEEE 34th International Conference on Distributed Computing Systems*. IEEE, Madrid, Spain, June 30-July 3, 2014, pp. 535–544.

[125] T. A. Runkler and H. Krcmar, "Stream processing on demand for lambda architectures," in *Proc. of 12th European Workshop on Computer Performance Engineering, Computer Performance Engineering*, ser. Lecture Notes in Computer Science, 2015, vol. 9272, pp. 243–257.

[126] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[127] L. George, *HBase: the definitive guide*. O'Reilly Media, Inc., 2011.

[128] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. of 9th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, San Jose, CA, April, 2012, pp. 2–2.

[129] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets." *HotCloud*, vol. 10, pp. 10–10, 2010.

[130] D. Borthakur, "Hdfs architecture guide. hadoop apache project," 2008. [Online]. Available: http://hadoop. apache. org/common/docs/current/hdfs_design. pdf

[131] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: a platform for fine-grained resource sharing in the data center," in *Proc. of 8th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, Boston, MA, March 30-April 1, 2011, pp. 295–308.

[132] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proc. of 4th Annual Symposium on Cloud Computing*. ACM, Santa Clara, CA, USA, October, 2013, p. 5.

[133] H. Li, A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Tachyon: Reliable, memory speed storage for cluster computing frameworks," in *Proc. of 2014 ACM Symposium on Cloud Computing*. ACM, Seattle, WA, November, 2014, pp. 1–15.

[134] T. White, *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.

[135] T. Kwok, K. Smith, S. Lozano, and D. Taniar, "Parallel fuzzy c-means clustering for large data sets," in *Proc. of European Conference on Parallel Processing, Euro-Par 2002 Parallel Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 365–374.

[136] J. Beringer and E. Hullermeier, "Fuzzy clustering of parallel data streams," *Advances in Fuzzy Clustering and Its Application*, pp. 333–352, 2007.

[137] Y. Zhang, S. Chen, Q. Wang, and G. Yu, "i^2mapreduce: Incremental mapreduce for mining evolving big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1906–1919, 2015.

[138] Y. Zhang, S. Cheen, Q. Wang, and G. Yu, "i^2mapreduce: Incremental mapreduce for mining evolving big data," *arXiv preprint arXiv:1501.04854*.

[139] N. M. Luscombe, D. Greenbaum, M. Gerstein *et al.*, "What is bioinformatics? a proposed definition and overview of the field," *Methods of Information in Medicine*, vol. 40, no. 4, pp. 346–358, 2001.

[140] S. Vipsita and S. K. Rath, "Two-stage approach for protein superfamily classification," *Computational Biology Journal*, vol. 2013, 2013.

[141] C. Wu, G. Whitson, J. McLarty, A. Ermongkonchai, and T. C. Chang, "Protein classification artificial neural system," *Protein Science*, vol. 1, no. 5, pp. 667–677, 1992.

[142] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu, "New techniques for extracting features from protein sequences," *IBM Systems Journal*, vol. 40, no. 2, pp. 426–441, 2001.

[143] C. S. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: a string kernel for svm protein classification," in *Proc. of 7th Pacific Symposium on Biocomputing*, vol. 7, no. 7, Lihue, Hawaii, USA, January, 2002, pp. 566–575.

[144] S. Bandyopadhyay, "An efficient technique for superfamily classification of amino acid sequences: feature extraction, fuzzy clustering and prototype selection," *Fuzzy Sets and Systems*, vol. 152, no. 1, pp. 5–16, 2005.

[145] E. G Mansoori, M. J. Zolghadri, S. D. Katebi, H. Mohabatkar, R. Boostani, and M. H. Sadreddini, "Generating fuzzy rules for protein classification," *Iranian Journal of Fuzzy Systems*, vol. 5, no. 2, pp. 21–33, 2008.

[146] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "Protein superfamily classification using fuzzy rule-based classifier," *IEEE Transactions on NanoBioscience*, vol. 8, no. 1, pp. 92–99, 2009.

[147] C. Caragea, A. Silvescu, and P. Mitra, "Protein sequence classification using feature hashing," *Proteome Science*, vol. 10, no. 1, p. 1, 2012.

[148] C. Yu, R. L. He, and S. S. T. Yau, "Protein sequence comparison based on k-string dictionary," *Gene*, vol. 529, no. 2, pp. 250–256, 2013.

[149] M. J. Iqbal, I. Faye, B. B. Samir, and A. Md Said, "Efficient feature selection and classification of protein sequence data in bioinformatics," *The Scientific World Journal*, vol. 2014, 2014.

[150] M. Dayhoff, R. Schwartz, and B. Orcutt, "22 a model of evolutionary change in proteins," in *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation Silver Spring, MD, 1978, vol. 5, pp. 345–352.

[151] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

[152] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[153] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.

[154] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[155] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[156] G. Frahling and C. Sohler, "Coresets in dynamic geometric data streams," in *Proc. of Thirty-Seventh Annual ACM Symposium on Theory of Computing*. ACM, Hunt Valley, MD, USA, May, 2005, pp. 209–217.

[157] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," *Large Scale Kernel Machines*, pp. 301–320, 2007.

[158] A. D. Gordon, "Cluster validation," in *Data science, Classification, and Related Methods*. Springer, 1998, pp. 22–39.

[159] W. Wang and Y. Zhang, "On fuzzy cluster validity indices," *Fuzzy Sets and Systems*, vol. 158, no. 19, pp. 2095–2117, 2007.

[160] J. Xiao, Y. Yan, J. Zhang, and Y. Tang, "A quantum-inspired genetic algorithm for k-means clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4966–4973, 2010.

[161] N. Bharill and A. Tiwari, "Enhanced cluster validity index for the evaluation of optimal number of clusters for fuzzy c-means algorithm," in *Proc. of 2014 IEEE International Conference on Fuzzy Systems*. IEEE, Beijing, China, July, 2014, pp. 1526–1533.

[162] O. P. Patel, N. Bharill, and A. Tiwari, "A quantum-inspired fuzzy based evolutionary algorithm for data clustering," in *Proc. of 2015 IEEE International Conference on Fuzzy Systems*. IEEE, Istanbul, Turkey, August, 2015, pp. 1–8.

[163] M. Duranton, D. Black-Schaffer, K. De Bosschere, and J. Maebe, "The hipeac vision for advanced computing in horizon 2020," 2013.

[164] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: the next frontier for innovation, competition, and productivity," *McKinsey Global Institute, Tech. Rep.*, no. 9341321, pp. 1–137, May 2011.

[165] M. Wang, W. Zhang, W. Ding, D. Dai, H. Zhang, H. Xie, L. Chen, Y. Guo, and J. Xie, "Parallel clustering algorithm for large-scale biological data sets," *PloS one*, vol. 9, no. 4, p. e91315, 2014.

[166] M. Cao Cueto, "Contribución a las metodologías de estimación de demanda de tráfico de internet mediante la caracterización de perfiles de usuario," Ph.D. dissertation, Telecomunicacion, 2015.

[167] V. Schwammle and O. N. Jensen, "A simple and fast method to determine the parameters for fuzzy c–means cluster analysis," *Bioinformatics*, vol. 26, no. 22, pp. 2841–2848, 2010.

[168] S. Vipsita and S. K. Rath, "Protein superfamily classification using adaptive evolutionary radial basis function network," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 04, p. 1250026, 2012.

[169] B. Chaturvedi and N. Patil, "A novel semi-supervised approach for protein sequence classification," in *Proc. of 2015 IEEE International Advance Computing Conference*. IEEE, Bangalore, India, June, 2015, pp. 1158–1162.

[170] J. Gubbi, D. T. Lai, M. Palaniswami, and M. Parker, "Protein secondary structure prediction using support vector machines and a new feature representation," *International Journal of Computational Intelligence and Applications*, vol. 6, no. 04, pp. 551–567, 2006.

[171] P. A. Devijver and J. Kittler, *Pattern recognition: a statistical approach*. Prentice-Hall London, 1982, vol. 761.

[172] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[173] D. L. Gray and A. N. Michel, "A training algorithm for binary feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 176–194, 1992.

[174] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[175] M. M. Mano, C. R. Kime, T. Martin *et al.*, *Logic and computer design fundamentals*. Prentice Hall, 2008, vol. 3.

[176] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane *et al.*, "Uniprot: the universal protein knowledgebase," *Nucleic Acids Research*, vol. 32, no. 1, pp. D115–D119, 2004.

[177] M. Tlili and T. M. Hamdani, "Big data clustering validity," in *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*. IEEE, 2014, pp. 348–352.

[178] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Multiple kernel fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 1, pp. 120–134, 2012.

[179] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[180] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition," in *Proc. of Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*. Citeseer, Istanbul, Turkey, June, 1996.

[181] M. Keith, A. Jameson, W. Van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. Bates, N. Bhat, M. Burgay *et al.*, "The high time resolution universe pulsar survey–i. system configuration and initial discoveries," *Monthly Notices of the Royal Astronomical Society*, vol. 409, no. 2, pp. 619–627, 2010.

[182] K. Wennerberg, K. L. Rossman, and C. J. Der, "The ras superfamily at a glance," *J cell Sci*, vol. 118, no. 5, pp. 843–846, 2005.

[183] S. N. Vinogradov, D. Hoogewijs, X. Bailly, K. Mizuguchi, S. Dewilde, L. Moens, and J. R. Vanfleteren, "A model of globin evolution," *Gene*, vol. 398, no. 1, pp. 132–142, 2007.

[184] C. I. Branden *et al.*, *Introduction to protein structure*. Garland Science, 1999.

[185] S. K. Hanks and T. Hunter, "Protein kinases 6. the eukaryotic protein kinase superfamily: kinase (catalytic) domain structure and classification." *The FASEB journal*, vol. 9, no. 8, pp. 576–596, 1995.

[186] M. A. Wouters, K. Liu, P. Riek, and A. Husain, "A despecialization step underlying evolution of a family of serine proteases," *Molecular cell*, vol. 12, no. 2, pp. 343–354, 2003.

Appendix A

UCI Datasets

The University of California, Irvine (UCI) Machine Learning Repository is a well-known resource to data scientists [154], As of 2016, it holds over 349 pre-processed datasets for experimentation and testing of machine learning algorithms. The datasets taken from this repository for the experimentation are discussed below.

IRIS

The well-known Fishers IRIS dataset is a multivariate dataset for discriminant analysis [179]. It consists of 150 samples belongs to 3 classes of IRIS flowers and four features were measured corresponding to each sample. The classes are IRIS Setosa, IRIS Versicolour and IRIS Virginica. The features are sepal length, sepal width, petal length, and petal width. The data set contains 50 instances of each of the three classes.

WINE

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivators and consists of 178 samples. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

VEHICLE

This data was originally gathered at the TI in 1986-87 by JP Siebert consists of 946 samples belongs to four classes. It was partially financed by Barr and

Stroud Ltd. The original purpose was to find a method of distinguishing 3D objects within a 2D image by application of an ensemble of shape feature extractors to the 2D silhouettes of the objects. Measures of shape features extracted from example silhouettes of objects to be discriminated were used to generate a classification rule tree by means of computer induction.

SEED

The dataset comprised of kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment.

GLASS

The dataset consists of 214 samples, for each sample 10 features are measured. All the samples are divided into 6 classes. The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence, if it is correctly identified.

BUPA

The dataset consist of 345 samples with 6 features are measured for each sample. The first 5 features are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption while the last attribute includes daily alcohol consumption. Each sample constitutes the record of a single male individual.

DIABETES

The PIMA INDIAN DIABETES dataset consists of 768 data samples that belong to only female patients suffering from diabetes. It has 8 numerical features per sample. Each sample contains a label which indicates the class of the sample. The dataset has two classes where the first class is labeled as "negative to diabetes" and the second one is labeled as "positive to diabetes".

Page Blocks Classification

The dataset comprises of 5473 samples comes from 54 distinct documents. Each observation concerns one block. It consists of 10 numeric features. The problem consists of classifying all the blocks of the page layout of a document that has been detected by a segmentation process. This is an essential step in document analysis in order to separate text from graphics areas. Indeed, the samples in the dataset are divided into five classes are: text, horizontal line, picture, vertical line, and graphics.

Pen-Based Recognition of Handwritten Digits

Pendigits is a collection of 10,992 handwritten numerals collected from 44 different writers. The handwritten numerals were plotted on an x and y coordinate grid. The 16 features are 8 (x, y) coordinates from the handwritten numeral. The coordinates were spatially resampled to be separated by an equal arc-length. The values are integers ranging from 0 to 100. The dataset was clustered into ten classes, corresponding to numerals from 0 to 9 [180].

HTRU2

The HTRU2 is a dataset which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [181]. The pulsar is a type of star, of considerable scientific interest. Candidate must be classified into pulsar and non-pulsar classes to aid discovery. The dataset consists of total 17898 samples of pulsar candidates out of which it contains 16,259 spurious examples caused by RFI/noise and 1,639 real pulsar examples. Each candidate is described by 8 continuous variables and a single class variable.

EEG Eye State

The dataset consists of total 14980 samples, where each sample is represented by 14 EEG values. All the samples are collected from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. Each sample is represented by a label or a value indicating the eye state. The eye state was detected via a camera during the

EEG measurement and added later manually to the file after analyzing the video frames. The label '1' indicates the eye-closed state and '0' the eye-open state.

Appendix B

Validation Methods

In machine learning field, it is common to partition the dataset into two separate sets: a training set and a testing set. To evaluate the generalizability of our approach and to compare our work with existing work in literature, we divided the training and testing data into four different partitions. The reason for dividing training and testing samples into different partitions is that we want to measure the quality of our proposed algorithms with different amount of training samples. The validation methods used for experimentation are discussed below.

50-50 training-testing partition

In standard 50-50 methodology, half of the samples are used for training the classifier and the rest for testing.

60-40 training-testing partition

In 60-40 training-testing methodology, 60% of the samples are used for training the classifier and the rest 40% for testing.

70-30 training-testing partition

In 70-30 training-testing methodology, 70% of the samples are used for training the classifier and the rest 30% for testing.

10-fold cross validation

In 10-fold cross validation technique entire dataset is divided into ten blocks of approximately equal size. The 90% of data is used to train the model and the

rest 10% for testing. This process is repeated 10 times, with a different data block left out for testing every time.

Appendix C

Benchmark Protein Database

The UniProt Knowledgebase is a large resource of protein sequences and associated detailed annotation [176]. The database contains over 60 million sequences, of which over half a million sequences have been curated by experts who critically review experimental and predicted data for each protein. We have created a benchmark protein database composed of variable length protein sequences of four superfamilies, i.e., Ras, Globin, Trypsin, and Kinase collected from a publicly available database, i.e., UniProtKB. The details of these superfamilies used for the experimentation are discussed below.

Ras superfamily

The Ras superfamily is a protein superfamily of small guanosine triphosphatases (GTPases) comprises of 154 human members [182]. The Ras superfamily is involved in signal transduction, the regulation of gene expression, cell proliferation, survival, and differentiation. The Ras oncogene proteins are the founding members of this family, which is divided into five major branches on the basis of sequence, structure, and functional similarities. The five major branches are Ras, Rho, Ran, Rab, and Arf. The Ras family itself is further divided into 6 subfamilies: Ras, Ral, Rit, Rap, Rheb, Rad, and Rit.

Globin superfamily

The globins are a superfamily of heme-containing globular proteins, involved in binding and/or transporting oxygen. They belong to a very large and well-studied family that is widely distributed in many organisms [183]. Globins

have evolved from a common ancestor and can be divided into three groups: single-domain globins, and two types of chimeric globins, flavo-haemoglobins and globin-coupled sensors. Globin superfamily members share a common three-dimensional fold [184]. This 'globin fold' typically consists of eight alpha helices. Since the globin fold contains only helices, it is classified as an all-alpha protein fold. Although the fold of the globin superfamily is highly evolutionarily conserved, the sequences that form the fold can have as low as 16% sequence identity.

Kinase superfamily

The protein kinases are one of the largest superfamilies of homologous proteins and genes. Within this family, there are now hundreds of different members whose sequences are known. They are related by virtue of their kinase domains (also known as catalytic domains), which consist of approximately 250-300 amino acid residues. Although there is a rich diversity of structures, regulation modes, and substrate specificities among the protein kinases, there are also common structural features. There are two main subdivisions within the superfamily: the protein-serine/threonine kinases and the protein-tyrosine kinases [185].

Trypsin superfamily

Trypsin is a serine protease from the PA clan superfamily, found in the digestive system of many vertebrates, where it hydrolyses proteins. Trypsin is formed in the small intestine when its proenzyme form, the trypsinogen produced by the pancreas, is activated. The serine proteinases are by far the best understood of all classes of enzymes. This is especially true for the serine proteinases of the trypsin superfamily. The depth knowledge on the trypsin superfamily comes from the detailed structures of several native mammalian and bacterial enzymes and of their complexes with molecules representing analogues of the various intermediate stages of the catalytic pathways [186].

Appendix D

Genome Database

To work on a real-life Big Data classification problem, a genome database of size 80 GB is collected from the Directorate of Soybean Research (DSR), Indore under Indian Council of Agricultural Research. This database is composed of proteins of various plant genomes which is used for the analysis. Over twenty-five sequenced plant genomes protein data were included in this analysis such as staple grain crops, maize (corn), rice, soybean, and wheat; fiber crops like cotton, hemp, and flax; fruits and vegetables, including apple, watermelon, tomato, strawberry, potato, cucumber, grape, and Chinese cabbage; and crops that are primarily important in developing countries such as the pigeon pea. The details of the genome database used in the experimental study are discussed below.

2S Albumin

2S albumin storage proteins are becoming of increasing interest in nutritional and clinical studies as they have been reported as major food allergens in SEED of many mono- and di-cotyledonous plants. As storage proteins, they are deposited in protein bodies of developing SEED and are utilized by the plant as a source of nutrients during subsequent germination and seedling growth. Recent findings have demonstrated that 2S albumins can also play a protective role in plants as defensive weapons against fungal attack. In addition to their physiological role in plants, these small globular proteins are becoming of increasing interest in nutritional and clinical studies.

β-conglycinin

The major storage proteins of soybeans are -conglycinin, or 7S globulins, and glycinin, or 11S globulins. Soybean protein has long been recognized as a source of dietary allergens for humans and animals with -conglycinin being the major allergen. -conglycinin is folded and assembled into trimers in the endoplasmic reticulum and accumulated into protein storage vacuoles.

Jasmonate O-methyltransferase

Jasmonate O-methyltransferase is an enzyme that catalyzes the chemical reaction. This enzyme belongs to the family of transferases, specifically those transferring one-carbon group methyltransferases. This enzyme is also called jasmonic acid carboxyl methyltransferase.

NBS-LRR

The most common disease resistance genes cloned to date are those belonging to the NBS-LRR family, named after the domains they typically contain: the nucleotide binding sites (NBS) and the leucine-rich repeat (LRR). This highly conserved gene family has structural and functional homology to the mammalian nucleotide-binding oligomerization domain (NOD)-LRR protein family, which functions in inflammatory and immune responses.

Cyclophilins

Plant cyclophilins were first identified in 1990 with the isolation of cyclophilin cDNA sequences from tomato (Lycopersicon esculentum), maize (Zea mays), and oilseed rape (Brassica napus). Cyclophilins are ubiquitous proteins present in all subcellular compartments, which are involved in a wide variety of processes including protein trafficking and maturation, apoptosis, RNA processing and spliceosome assembly.

Heat shock proteins (HSP)

Heat shock proteins (HSP) are a family of proteins that are produced by cells in response to exposure to stressful conditions. They were first described in relation to heat shock,but are now known to also be expressed during other

stresses including exposure to cold, UV light, and during wound healing or tissue remodeling.

Plant ABC Transporters

The ATP binding cassette (ABC) superfamily is a large, ubiquitous and diverse group of proteins, most of which mediate transport across biological membranes. ABC transporters have been shown to function not only as ATP-dependent pumps, but also as ion channels and channel regulators.

The dehydration responsive elements (DREBs)

Plants have evolved intricate mechanisms to respond and adapt to a wide variety of biotic and abiotic stresses in their environment. DREB family regulatory genes are involved in many different pathways, including genes related to cold, drought, high salinity, heavy metals, and abscisic acid (ABA). DREB genes play an important role in the ABA-independent stress-tolerance pathways that induce the expression of various stress-responsive genes in plants. ABA is an important plant hormone that plays a regulatory role in many physiological processes in plants, such as embryo maturation, seed development, and the activation of stress-responsive genes.

Arah1

Arah1 is a seed storage protein from Arachis hypogaea (peanuts). It is a heat stable 7S vicilin-like globulin with a stable trimeric form that comprises 12-16% of the total protein in peanut extracts. Arah1 is known because sensitization to it was found in 95% of peanut-allergic patients from North America. In spite of this high percentage, peanut-allergic patients of European populations have fewer sensitizations to Ara h 1. No treatment is currently available, avoidance is the only option for peanut-allergic individuals.

Arah3

Food allergy is a worldwide health problem. It affects approximately 5% of young children and 3% to 4% of adults in westernized countries. Among them, peanut is one of the most allergenic. Peanut allergy affects many individuals

and its prevalence is increasing rapidly. Arah3 is a seed storage protein and belongs to the legumin (11S) family. It is recognized by 50% of peanut-allergic individuals and also functions as a trypsin inhibitor