

Fall 2012

Adaptive grid based localized learning for multidimensional data

Sheetal Saini

Follow this and additional works at: <https://digitalcommons.latech.edu/dissertations>

 Part of the [Databases and Information Systems Commons](#)

**ADAPTIVE GRID BASED LOCALIZED LEARNING
FOR MULTIDIMENSIONAL DATA**

by

Sheetal Saini, B. Eng.

A Dissertation Presented in Partial Fulfillment
of the Requirements of the Degree
Doctor of Philosophy

**COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY**

November 2012

UMI Number: 3534292

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3534292

Published by ProQuest LLC 2012. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

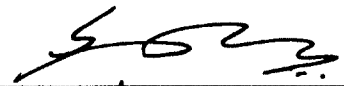
9/21/2012

Date

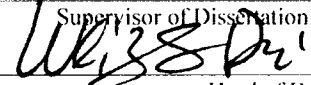
We hereby recommend that the dissertation prepared under our supervision
by Sheetal Saini

entitled Adaptive Grid Based Localized Learning for Multidimensional Data

be accepted in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in Computational Analysis and Modeling



Supervisor of Dissertation Research



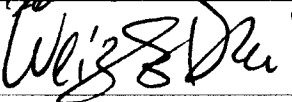
Head of Department

Computational Analysis and Modeling

Department

Recommendation concurred in:

Milang W. Thompson / S.D.



Advisory Committee

Karna Saha

Approved:

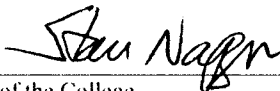


Director of Graduate Studies

Approved:



Dean of the Graduate School



Dean of the College

ABSTRACT

Rapid advances in data-rich domains of science, technology, and business has amplified the computational challenges of “Big Data” synthesis necessary to slow the widening gap between the rate at which the data is being collected and analyzed for knowledge. This has led to the renewed need for efficient and accurate algorithms, framework, and algorithmic mechanisms essential for knowledge discovery, especially in the domains of clustering, classification, dimensionality reduction, feature ranking, and feature selection. However, data mining algorithms are frequently challenged by the sparseness due to the high dimensionality of the datasets in such domains which is particularly detrimental to the performance of unsupervised learning algorithms.

The motivation for the research presented in this dissertation is to develop novel data mining algorithms to address the challenges of high dimensionality, sparseness and large volumes of datasets by using a unique grid-based localized learning paradigm for data movement clustering and classification schema. The grid-based learning is recognized in data mining as these algorithms are inherently efficient since they reduce the search space by partitioning the feature space into effective partitions. However, these approaches have not been successfully devised for supervised learning algorithms or sparseness reduction algorithm as they require careful estimation of grid sizes, partitions and data movement error calculations. Grid-based localized learning algorithms can scale well with an increase in dimensionality and the size of the datasets.

To fulfill the goal of designing and developing learning algorithms that can handle data sparseness, high data dimensionality, and large size of data, in a concurrent manner to avoid the feature selection biases, a set of novel data mining algorithms using grid-based localized learning principles are developed and presented. The first algorithm is a unique computational framework for feature ranking that employs adaptive grid-based data shrinking for feature ranking. This method addresses the limitations of existing feature ranking methods by using a scoring function that discovers and exploits dependencies from all the features in the data. Data shrinking principles are established and metricized to capture and exploit dependencies between features. The second core algorithmic contribution is a novel supervised learning algorithm that utilizes grid-based localized learning to build a nonparametric classification model. In this classification model, feature space is divided using uniform/non-uniform partitions and data space subdivision is performed using a grid structure which is then used to build a classification model using grid-based nearest-neighbor learning. The third algorithm is an unsupervised clustering algorithm that is augmented with data shrinking to enhance the clustering performance of the algorithm. This algorithm addresses the limitations of the existing grid-based data shrinking and clustering algorithms by using an adaptive grid-based learning. Multiple experiments on a diversified set of datasets evaluate and discuss the effectiveness of dimensionality reduction, feature selection, unsupervised and supervised learning, and the scalability of the proposed methods compared to the established methods in the literature.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author SHEETAL SAINI

Date 09/21/2012

DEDICATION

To my family and elders for their sacrifices, support, patience, and endless prayers for me.

TABLE OF CONTENTS

ABSTRACT.....	iii
DEDICATION.....	vi
LIST OF TABLES.....	xii
LIST OF FIGURES.....	xiv
ACKNOWLEDGEMENTS.....	xvii
CHAPTER 1 INTRODUCTION.....	1
1.1 Knowledge Discovery in Databases.....	2
1.2 Data Mining.....	4
1.3 Learning Techniques.....	5
1.3.1 Unsupervised Learning.....	6
1.3.2 Supervised Learning.....	6
1.4 Localized Learning.....	7
1.4.1 Nearest-Neighbor Learning.....	7
1.4.2 Grid-Based Nearest-Neighbor Learning.....	8
1.5 Dissertation Organization.....	9
CHAPTER 2 RELATED RESEARCH.....	12
2.1 Grid-Based Localized Learning.....	13
2.2 Data Shrinking.....	14
2.2.1 Point-Based Approach.....	16
2.2.2 Grid-Based Approach.....	16

2.3 Feature Selection and Ranking	16
2.4 Classification	17
2.5 Clustering.....	19
2.5.1 Partitioning-Based Clustering.....	19
2.5.2 Density-Based Clustering	19
2.5.3 Hierarchical Clustering	20
2.5.3.1 Agglomerative Hierarchical Clustering	20
2.5.3.2 Divisive Hierarchical Clustering.....	21
2.5.4 Grid-Based Clustering	21
2.5.4.1 Uniform Grid-Based Clustering.....	22
2.5.4.2 Non-Uniform Grid-Based Clustering	22
2.5.5 Data Shrinking Based Clustering.....	22
2.6 Conclusion	24
CHAPTER 3 PRELIMINARIES OF GRID-BASED LOCALIZED LEARNING.....	25
3.1 Notations.....	25
3.2 Formal Definitions.....	26
CHAPTER 4 GRID-BASED LOCALIZED LEARNING FOR DATA PREPROCESSING.....	35
4.1 Data Preprocessing	36
4.2 Data Sparseness	38
4.3 Research Motivation	39
4.3.1 Limitations of Existing Techniques	39
4.3.2 Advantages of Non-Uniform Grid	40
4.4 Experimental Study.....	41

4.4.1 Datasets	41
4.4.2 Effect of Sparseness	42
4.4.3 Comparative Study.....	43
4.4.3.1 Comparison of Partitioning Methods.....	43
4.4.3.2 Comparison of Shrinking Methods.....	46
4.5 Conclusion	50
CHAPTER 5 GRID-BASED LOCALIZED LEARNING FOR FEATURE RANKING	51
5.1 Research Motivation	52
5.2 Problem Statement	53
5.3 Methodology.....	53
5.3.1 Data Preprocessing.....	54
5.3.2 Adaptive Grid Generation.....	54
5.3.3 Data Shrinking	55
5.3.3.1 Data Movement Model	56
5.3.3.2 Data Shrinking Process	57
5.3.4 Feature Ranking Method.....	58
5.4 Results and Discussions.....	60
5.4.1 Datasets	60
5.4.2 Validation.....	61
5.4.3 Experiments	61
5.5 Conclusion	68
CHAPTER 6 GRID-BASED LOCALIZED LEARNING FOR CLASSIFICATION.....	69
6.1 Research Motivation	70

6.2 Problem Statement	70
6.3 Methodology	70
6.3.1 Data Preprocessing.....	71
6.3.2 Grid Generation	71
6.3.2.1 Uniform Grid Generation.....	71
6.3.2.2 Adaptive Grid Generation.....	72
6.3.3 Training Phase	74
6.3.4 Test Phase	76
6.4 Results and Discussions.....	77
6.4.1 Datasets	78
6.4.2 Validation.....	79
6.4.3 Experiments	80
6.4.3.1 Scalability Analysis	80
6.4.3.2 Comparative Analysis.....	87
6.4.4 Time Complexity Analysis	89
6.5 Conclusion	90
CHAPTER 7 GRID-BASED LOCALIZED LEARNING FOR CLUSTERING.....	92
7.1 Research Motivation	93
7.2 Problem Statement.....	93
7.3 Methodology.....	94
7.3.1 Data Preprocessing.....	95
7.3.2 Adaptive Grid Generation	95
7.3.2.1 Finding Micro-Partitions.....	96

7.3.2.2 Data Transformation	97
7.3.2.3 MOSAH Partitioning	98
7.3.2.4 Algorithmic Description	99
7.3.3 Adaptive Grid-Based Shrinking.....	100
7.3.3.1 Ranking Neighboring Grid Cells	100
7.3.3.2 Data Movement Model	102
7.3.3.3 Data Shrinking Process	103
7.3.4 Adaptive Grid-Based Clustering.....	105
7.4 Results and Discussion	107
7.4.1 Datasets	107
7.4.2 Validation.....	108
7.4.3 Experiments	109
7.4.3.1 Scalability Analysis	110
7.4.3.2 Comparative Analysis.....	114
7.4.4 Time Complexity Analysis	129
7.5 Conclusion	130
CHAPTER 8 CONCLUSIONS	132
8.1 Contribution to Grid-Based Supervised Learning	132
8.2 Contribution to Grid-Based Unsupervised Learning	133
REFERENCES	134

LIST OF TABLES

Table 5.1: Common Top Ranked Features	63
Table 5.2: Comparison of F-measure for Neural Network	64
Table 5.3: Comparison of F-measure for PART	65
Table 5.4: Comparison of F-measure for Logistic Regression	66
Table 5.5: Comparison of Avg. Precision, Recall, Accuracy for PART	66
Table 5.6: Comparison of Avg. Precision, Recall, Accuracy for Logistic Regression...	67
Table 5.7: Comparison of Avg. Precision, Recall, Accuracy for Neural Network.....	68
Table 7.1: Benchmark v/s Adaptive Shrinking Based Method on Wine Dataset.....	115
Table 7.2: CURE v/s Adaptive Shrinking Based Method on Wine Dataset.....	116
Table 7.3: DBSCAN v/s Adaptive Shrinking Based Method on Wine Dataset	116
Table 7.4: Benchmark v/s Adaptive Shrinking Based Method on Ecoli Dataset	117
Table 7.5: CURE v/s Adaptive Shrinking Based Method on Ecoli Dataset.....	118
Table 7.6: DBSCAN v/s Adaptive Shrinking Based Method on Ecoli Dataset	119
Table 7.7: Benchmark v/s Adaptive Shrinking Based Method on Protein Dataset	120
Table 7.8: CURE v/s Adaptive Shrinking Based Method on Protein Dataset.....	121
Table 7.9: DBSCAN v/s Adaptive Shrinking Based Method on Protein Dataset	122
Table 7.10: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset	123
Table 7.11: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset	124
Table 7.12: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset.....	124

Table 7.13: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset	125
Table 7.14: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset	126
Table 7.15: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset.....	127
Table 7.16: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset	128
Table 7.17: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset	129
Table 7.18: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset.....	129

LIST OF FIGURES

Figure 1.1: KDD Process	3
Figure 1.2: Data Mining as Confluence of Multiple Disciplines.....	5
Figure 1.3: Two-Dimensional Grid.....	8
Figure 1.4: Key Elements of This Dissertation.....	9
Figure 2.1: Grid-Based Localized Learning Paradigm.....	14
Figure 2.2: Data Shrinking Approaches.....	15
Figure 2.3: Types of Data Grids	22
Figure 3.1: A Two-Dimensional Uniform Grid.....	27
Figure 3.2: A Two-Dimensional Non-Uniform Grid.....	28
Figure 3.3: A Grid Cell Representation	28
Figure 4.1: Average Pairwise Euclidean Distance v/s Dimensions	42
Figure 4.2: \log_2 (Total Grid Cells/Non-Empty Grid Cells) v/s Dimensions.....	44
Figure 4.3: \log_2 (Total Grid Cells /Non-Empty Grid Cells) v/s Dimensions.....	45
Figure 4.4: \log_2 (Total Grid Cells/Non-Empty Grid Cells) v/s Dimensions.....	46
Figure 4.5: Cumulative Wavelet Entropy v/s Dimensions	47
Figure 4.6: Cumulative Energy v/s Dimensions	48
Figure 4.7: Cumulative Information Entropy v/s Dimensions.....	49
Figure 4.8: Average Execution Time v/s Dimensions	50
Figure 5.1: Data Shrinking Based Feature Ranking Framework.....	54
Figure 5.2: Adaptive Grid Generation Algorithm.....	55

Figure 5.3: Adaptive Data Shrinking Algorithm	57
Figure 5.4: Feature Ranking and Selection Algorithm	59
Figure 5.5: Comparison of Top Ranked Features	62
Figure 6.1: Variance-Based Partitioning Algorithm.....	73
Figure 6.2: Training Phase of the Grid-Based Classifier.....	75
Figure 6.3: Test Phase of the Grid-Based Classifier.....	77
Figure 6.4: Training Phase Execution Time v/s Dimensions	81
Figure 6.5: Training Phase Execution Time v/s Dataset Size.....	82
Figure 6.6: Average Execution Time/Sample v/s Dimensions.....	83
Figure 6.7: Average Execution Time/Sample v/s Dataset Size	83
Figure 6.8: Execution Time v/s Dimensions.....	84
Figure 6.9: Execution Time v/s Dataset Size.....	85
Figure 6.10: Average Execution Time/Sample v/s Dimensions.....	86
Figure 6.11: Average Execution Time/Sample v/s Dataset Size	86
Figure 6.12: Comparative Study on Letter Recognition Dataset.....	87
Figure 6.13: Comparative Study on Profile Correlation Feature Set.....	88
Figure 6.14: Comparative Study on Protein Structural Classification Dataset.....	89
Figure 7.1: Adaptive Shrinking Based Clustering Approach	94
Figure 7.2: MOSAH Partitioning of Micro-partitions	98
Figure 7.3: Data Adaptive Grid Generation Algorithm.....	99
Figure 7.4: A Two-Dimensional Grid with Cell ID's.....	101
Figure 7.5: Hierarchical Decomposition of Data Adaptive Partitions.....	103
Figure 7.6: Pseudo-code for Data Shrinking Algorithm.....	104

Figure 7.7: Grid-Based Hierarchical Clustering	105
Figure 7.8: Pseudo-code for Adaptive Grid-Based Clustering	106
Figure 7.9: Execution Time v/s Dataset Size (Analysis for Grid Generation Method).....	110
Figure 7.10: Execution Time v/s Dimensions (Analysis for Grid Generation Method).....	111
Figure 7.11: Execution Time v/s Dataset Size (Analysis for Data Shrinking Method).....	112
Figure 7.12: Execution Time v/s Dimensions (Analysis for Data Shrinking Method).....	112
Figure 7.13: Execution Time v/s Dataset Size (Analysis for Clustering Method)	113
Figure 7.14: Execution Time v/s Dimensions (Analysis for Clustering Method)	113

ACKNOWLEDGEMENTS

My dissertation has not only tested my skills and determination, but it has also tested the support and patience of my family, close friends and people around me. First of all, I would like to thank Dr. Sumeet Dua for providing his guidance and financial support throughout my PhD. I am grateful to the Louisiana Biomedical Research Network (LBRN), the Louisiana Alliance for Simulation-Guided Materials Applications (LA-SiGMA), National Science Foundation (NSF), and National Institutes of Health (NIH) for providing financial support for my education and research. I am only able to achieve my goals through the support and patience of my whole family. I am indebted to them for their support and patience throughout my PhD, especially my grandmother's and my mother's endless prayers for me. Finally, I would like to thank my close friends who extended their valuable support whenever possible. I would have not finished it without their support.

CHAPTER 1

INTRODUCTION

The common characteristics of contemporary datasets are multi dimensionality, sparseness, and the large size of the data. These characteristics are the main motivation behind the development of novel algorithms and frameworks for automated and sophisticated data mining systems that search nontrivial, previously unknown, and potentially useful knowledge from the data. Many researchers and scientists have developed automated systems that address these problems. As a result, ample literature on these problems and potential solutions are available. However, there is always a need to improve the existing algorithms, frameworks, and systems to achieve better performance and address the shortcomings of the existing data mining techniques.

Data mining techniques are commonly categorized based on the type of knowledge mined by these techniques. The most common data mining techniques are classification, and clustering. Classification is used to build models based on the data and known class labels that can describe data classes or groups [1, 2]. It predicts categorical class labels based on known examples. Therefore, it is also referred to as supervised learning. There are ample classification techniques, such as decision tree classifier, Bayesian classifier, rule based classifier, neural network classifier, support vector machine, k-nearest-neighbor classifier, and others. Unlike classification, clustering and unsupervised learning does not rely on predefined classes and class-labeled training

examples. For this reason, clustering is a form of learning by observation, rather than learning by examples.

The algorithms presented in this dissertation are created using the grid-based localized learning paradigm of data mining for knowledge discovery. To explain these paradigms, the understanding of the knowledge discovery process, data mining, machine learning, and localized learning are critical. Therefore, the process of knowledge discovery in databases (KDD), data mining, which is the core of the KDD process, machine learning, and localized learning and grid-based localized learning paradigms are outlined and explained in this chapter.

1.1 Knowledge Discovery in Databases

The phrase knowledge discovery in databases commonly (KDD) refers to the process of extracting nontrivial, implicit, previously unknown, valid, potentially useful, and understandable patterns/knowledge from data in databases by applying data mining algorithms [1]. Knowledge discovery in databases (KDD) is an interactive and iterative process that involves many decisions made by the end user. Knowledge discovery in databases process includes data selection, data preprocessing, data transformation, data mining, and data evaluation/interpretation. All the steps involved in the KDD process are defined and discussed below. Figure 1.1 depicts the KDD process.

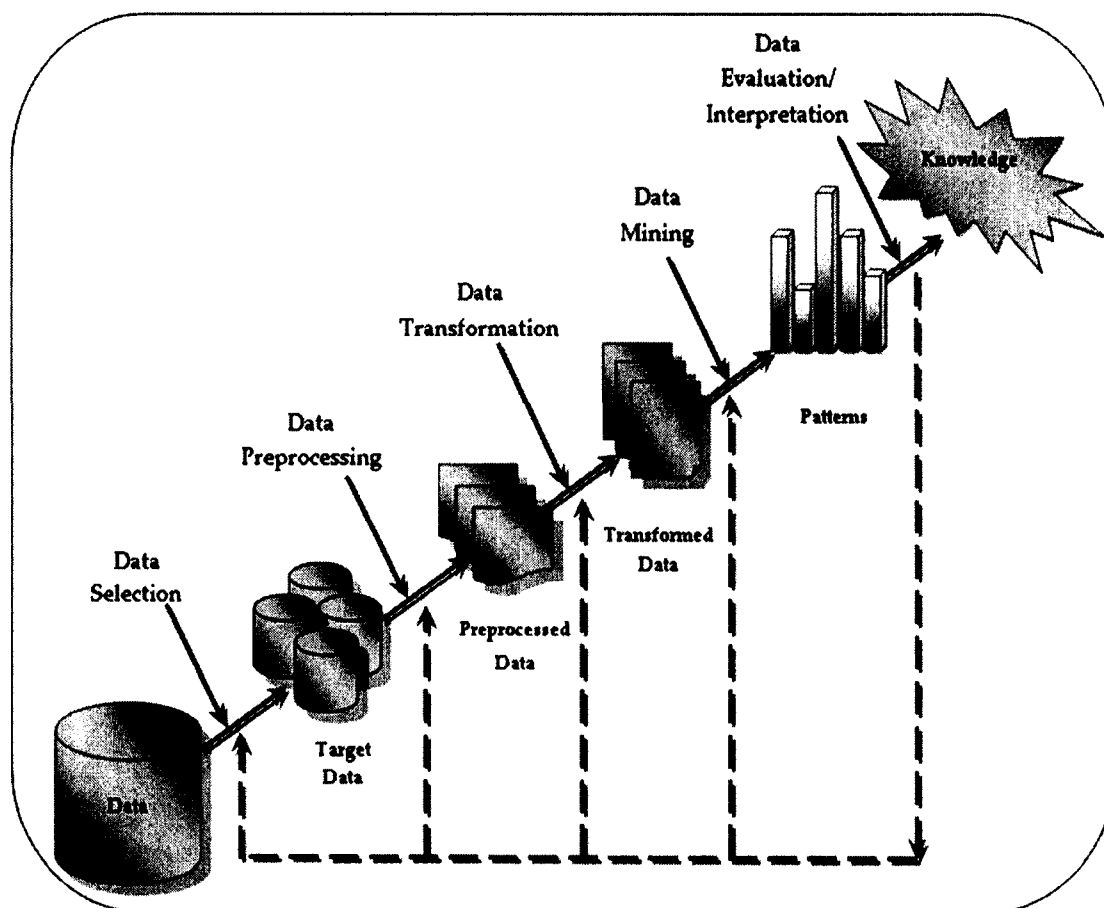


Figure 1.1: KDD Process

1. Data Selection: Data selection is the process of creating a target dataset on which knowledge discovery is to be performed. Extracting a target dataset refers to the selection of a subset of data attributes, data samples, or both attributes and samples that are relevant for the analysis task at hand [1].

2. Data Preprocessing: Data preprocessing is a data cleaning process, which involves operations such as removing noise, filling in missing values, and eliminating inconsistent data. It requires identification and selection of appropriate method for each operation.

3. Data Transformation: Data transformation is the process of converting data into the format that is most appropriate for relevant data mining tasks. Data

transformation includes data aggregation, data smoothing, data normalization, data generalization, and feature construction [1].

4. Data Mining: Data mining in the KDD process is a step that involves extracting patterns/knowledge of interest in a particular representational form by applying an appropriate data modeling technique. These data modeling techniques include association rule discovery, classification models, clustering models, and prediction models [1].

5. Data Evaluation/Interpretation: Data evaluation and interpretation is the process in which discovered patterns/knowledge is evaluated. This step also involves the interpretation of patterns through visualization or other means of representation.

1.2 Data Mining

Data mining is the process of extracting or mining interesting and useful patterns or knowledge from the given data [2]. In data mining, the term 'extraction of patterns or knowledge' refers to fitting a model to data, finding implicit structure from the data, or describing the data through a high level of abstraction [3]. There are two prevalent perspectives regarding data mining. The first perspective treats data mining as a synonym for knowledge discovery in databases (KDD), and the second perspective treats data mining as an essential step in the process of knowledge discovery in databases (see Figure 1.1). In both cases, data mining is an interdisciplinary field, and it is a confluence of multiple disciplines. Disciplines that contribute to data mining are database systems, statistics, machine learning, visualization, and information science [2]. It relies heavily on machine learning, pattern recognition, mathematics, and statistical techniques to find

patterns/knowledge from data [2]. Figure 1.2 depicts the interdisciplinary view of data mining.

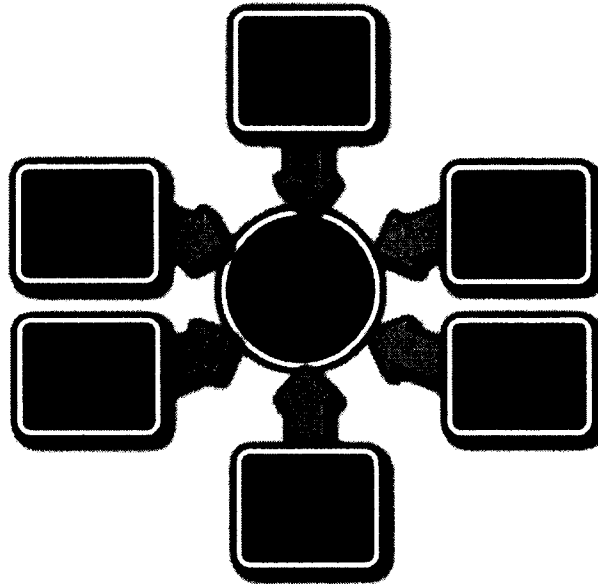


Figure 1.2: Data Mining as Confluence of Multiple Disciplines

As shown in Figure 1.2, data mining is the process of applying specific methods to extract interesting patterns/knowledge from the data [1].

1.3 Learning Techniques

Machine learning is a domain of artificial intelligence methods that are designed to automatically learn to recognize the evolving behavior of the system based on sample data. The term also refers to designing algorithms that optimize the performance criteria of the chosen mathematical model based on the input data [4]. These mathematical models can be predictive or descriptive. Predictive models are used to predict future outcomes, and descriptive models are used to gain knowledge about the data. Machine learning techniques can be broadly categorized into supervised learning and unsupervised

learning [4]. In subsections 1.3.1 and 1.3.2, supervised learning and unsupervised learning methods are explained.

1.3.1 Unsupervised Learning

Unsupervised learning is learning by observation, rather than learning by example. It does not rely on predefined classes and class-labeled training examples [2]. In unsupervised learning, the class label of each data point is not known. In some cases, the total number of classes to be learned may not be known in advance. The aim of the unsupervised learning is to identify patterns in the data that occur more often than others based on the structure of the data space [4, 5]. Commonly employed unsupervised learning techniques are clustering, subspace clustering, bi-clustering, and density estimation. The basic principle of all these techniques is to group the data into clusters such that data points within a cluster are very similar to each other but are very dissimilar to the data points in other clusters.

1.3.2 Supervised Learning

Supervised learning is learning by example. It relies on the knowledge about the class labels of each data point and the number of classes. Supervised learning is a two-step process. In the first step, a learning model is built using the predefined number of classes and class labels of each data point. This learning step is called the training phase. Each data point is assumed to belong to a predefined class which is determined by a class label attribute. The class label attribute is categorical, and each value serves as a class identifier [2]. The data points that are part of the training phase are collectively referred to as a training set and are selected from the given dataset. In the second step, the model learned in the first step is used to assign class labels to the data points that do not have any class label. This step is also called the testing phase. The data points that are part of

the testing phase are collectively referred to as the test set and are also selected from the given dataset for validation.

1.4 Localized Learning

Two commonly used learning techniques are known as parametric learning and nonparametric learning, respectively [4]. In parametric learning, a valid model is assumed for the whole input space, whereas in nonparametric learning no model is assumed. In nonparametric learning, there is no single global model, but local models are built based on the local neighborhood [4]. Therefore, a nonparametric learning strategy can also be referred to as 'localized learning.'

All the localized learning methods follow the same philosophy and can only be differentiated based on the similarity criteria of the neighborhood. Distance based nearest-neighbor learning is the most common form of neighborhood learning, but other methods such as grid-based nearest-neighbor learning and rule based nearest-neighbor learning are used in machine learning as well [2, 6, 7, 8, 9, 10]. Localized learning refers to the method of learning in which local models are learned or built based on a local neighborhood.

1.4.1 Nearest-Neighbor Learning

Nearest-neighbor learning is based on the intuition that an input data instance is more likely to be similar to input data instances that are in the neighborhood. 1-NN and k-NN are two common nearest-neighbor learning strategies. In the 1-NN nearest-neighbor method only one nearest-neighbor is identified, whereas, in the k-NN nearest-neighbor method the total 'k' numbers of nearest-neighbors are identified [5]. Nearest-neighbor learning is also referred to as a prototype method [5]. Nearest-neighbor learning

has been used for both supervised (k-NN classifier) and unsupervised (k-NN estimator) learning. Grid-based nearest-neighbor learning, an important aspect of the research presented in the first part of the dissertation, is explained below.

1.4.2 Grid-Based Nearest-Neighbor Learning

The idea of grid-based nearest-neighbor learning originates from a class of clustering algorithms known as grid-based clustering algorithms [6, 7, 8, 9, 10, 11, 12]. In grid-based clustering algorithms, initially, dimensions are divided into two or more partitions, and a grid structure is imposed on the feature space. This grid structure then divides the feature space into small cells called grid cells (see Figure 1.3). Next, each data sample is mapped onto the grid structure and assigned to a corresponding grid cell. Finally, these grid cells are used for clustering, and neighbors are identified by searching for adjacent non-empty grid cells. Figure 1.3 depicts a two-dimensional grid structure.

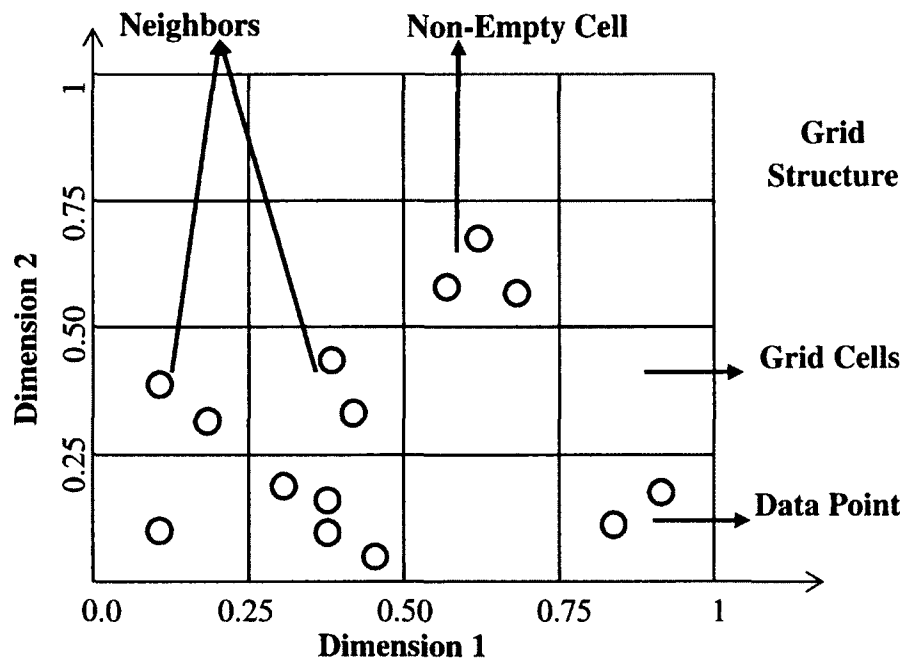


Figure 1.3: Two-Dimensional Grid

Thus, in grid-based nearest-neighbor learning, the definition of a neighborhood is based on the concept of grid cells, rather than individual data points.

1.5 Dissertation Organization

The remainder of the dissertation is further divided into seven more chapters. The organization and the outline of the remaining dissertation are as follows. A pictorial representation of the key elements of this dissertation is presented in Figure 1.4.

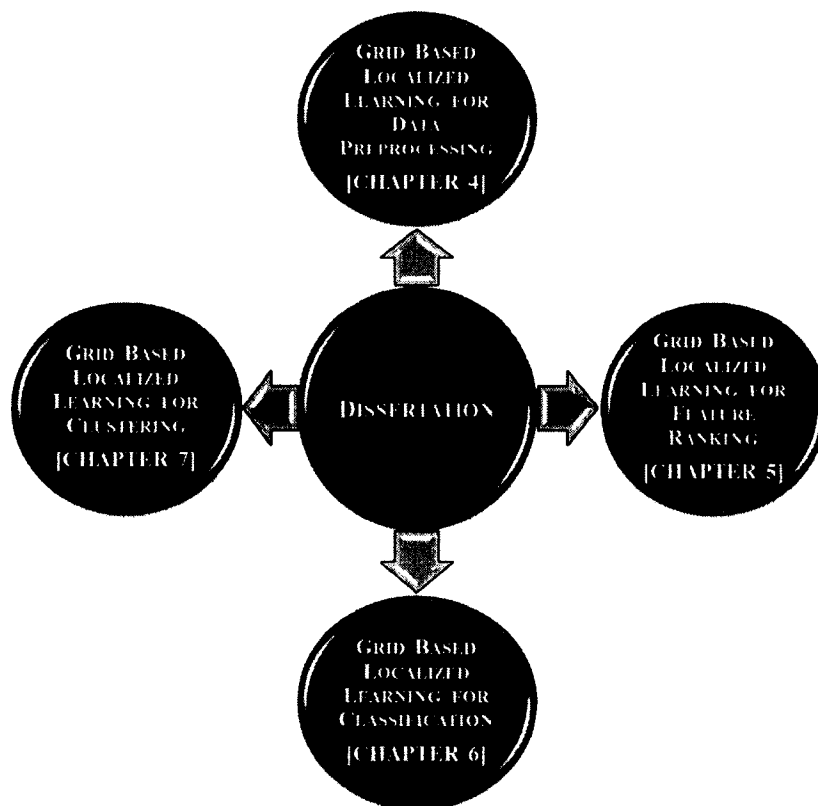


Figure 1.4: Key Elements of This Dissertation

Chapter 2: In Chapter 2, research related to the problem domain of this dissertation is presented. It includes discussion on pertinent literature review on data shrinking preprocessing, feature ranking, classification and clustering techniques.

Chapter 3: In Chapter 3, preliminaries of grid-based localized learning are presented. It includes description of notations that are used in subsequent chapters. It also includes formal definitions of various terminologies that are essential in understanding the concepts of grid-based localized learning paradigm.

Chapter 4: In Chapter 4, the need for data preprocessing and various methods of data preprocessing techniques are discussed. However, special emphasis is given to data shrinking preprocessing techniques and its need for sparseness reduction in multidimensional data. This chapter also includes experimental studies that demonstrate the benefits of the newly developed sparseness reduction technique presented in this dissertation.

Chapter 5: In Chapter 5, a feature ranking method is presented that uses the grid-based localized learning method. This chapter discusses research motivation, problem statement, and methodology. Experimental studies are also presented in which comparative studies of the existing and newly developed feature ranking methods are performed.

Chapter 6: In Chapter 6, a grid-based localized learning method is presented for classification. This chapter includes discussion on research motivation, problem statement and explains the developed grid-based classification framework. Finally, experimental studies are presented to compare the newly developed framework with existing methodology.

Chapter 7: In Chapter 7, grid-based data shrinking and clustering algorithm is presented. This chapter includes motivation and the problem statement for the research.

The developed methodology is also explained in detail which is further supported by experimental study conducted.

Chapter 8: In Chapter 8, the conclusions and future directions are presented. It also includes the outcomes of this dissertation.

CHAPTER 2

RELATED RESEARCH

Many data mining algorithms have been developed to address the challenges of data sparseness, the curse of dimensionality, and the large size of the data [2, 4, 5, 13, 14]. Many learning techniques have been developed to address these challenges. These learning techniques are categorized into parametric and nonparametric approaches [2, 4, 5, 15, 16, 17, 18]. In parametric learning approaches, a global model is built for all data samples at once. In nonparametric learning approaches, local models are built using the local neighborhood [4, 5]. Therefore, nonparametric approaches can also be referred to as localized learning approaches. Nonparametric techniques of data modeling have advantages over parametric techniques because of its simplicity [4, 5]. In the past, several approaches have been developed for data mining using both parametric and nonparametric learning models [2, 4, 5]. However, the focus of the research in this dissertation is on using grid-based localized learning techniques to address the challenges of data sparseness, the curse of dimensionality, and the large size of data in data mining. This chapter includes a discussion on the research related to clustering techniques , feature ranking techniques, data shrinking techniques, and classification techniques to provide the general idea of these techniques and demonstrate a need to develop grid-based localized learning techniques in these areas to address data mining challenges.

The remainder of the chapter is organized as follows. In Section 2.1, grid-based localized learning is explained and discussed. In Section 2.2, research related to data shrinking preprocessing, including existing grid-based shrinking approaches and non-grid/point-based approaches, is explained [13, 14, 20, 21, 22, 23, 24]. In Section 2.3, research related to feature selection and ranking is discussed. In 2.4, research related to classification techniques. In Section 2.5, research related to clustering techniques is discussed in general. However, special emphasis is given to grid-based clustering techniques and clustering techniques that are augmented with data preprocessing techniques to boost their performance. Finally, in Section 2.6, the conclusions of this chapter are presented.

2.1 Grid-Based Localized Learning

Grid-based learning algorithms are nonparametric learning algorithms. In these algorithms, a grid structure is imposed on the data space that divides it into smaller partitions called grid cells. Data is mapped in these grid cells which are then used to build local models using grid-based neighborhood learning [2]. In the past, grid-based localized learning has been used extensively for designing unsupervised learning algorithms such as clustering, subspace clustering, and data shrinking [25, 26, 27, 28, 29, 13, 14]. In this dissertation, the scope of grid-based localized learning is further expanded into grid-based data preprocessing techniques, such as data shrinking, grid-based supervised learning techniques, grid-based clustering techniques, and grid-based data shrinking and dimensionality reduction [30, 31]. Figure 2.1 depicts a schematic of a grid-based localized learning paradigm.

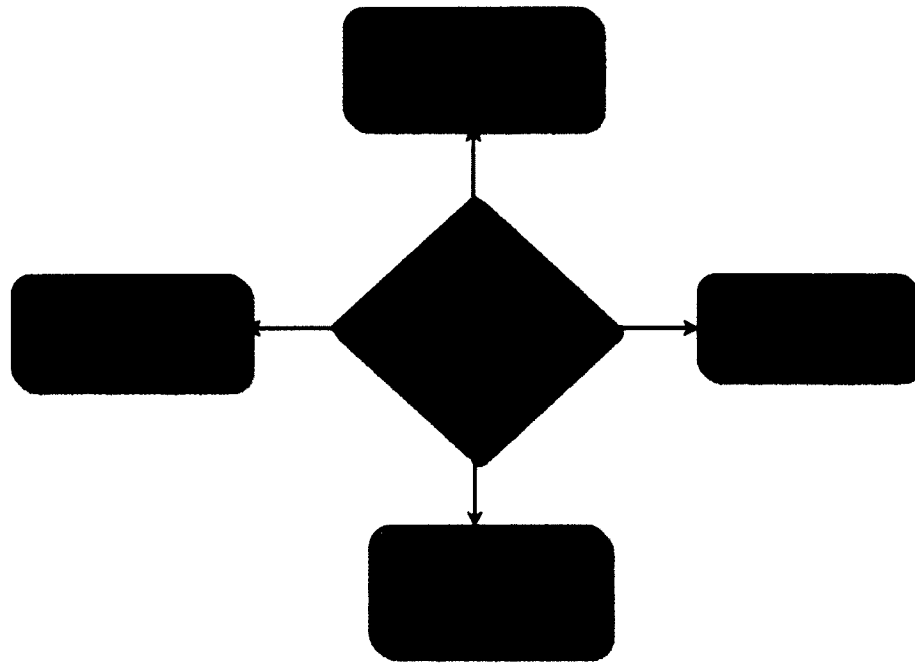


Figure 2.1: Grid-Based Localized Learning Paradigm

The schematic depicts the applicability of grid-based localized learning in the area of clustering, classification, data shrinking and dimensionality reduction techniques.

2.2 Data Shrinking

Data shrinking is a data preprocessing technique that is used to reduce the sparseness in a multidimensional dataset. The sparseness of the data increases as the number of dimensions increases [13, 14]. As a result, clusters of data points lack distinct boundaries, and the detection of clusters with better accuracies is severely affected. The data shrinking process utilizes the inherent characteristics of data distribution and outputs a more condensed and reorganized dataset [13, 14]. In the data shrinking process, the movement of data points is performed through the principle of data gravitation. Points are attracted by their surrounding neighbors and move toward the center of their natural clusters along the direction of the density gradient [20, 21, 22, 23, 24]. Furthermore, data shrinking approaches can be broadly categorized into grid-based approaches and non-

grid/point-based approaches. A pictorial representation of the categorization of data shrinking approaches is presented in Figure 2.2.

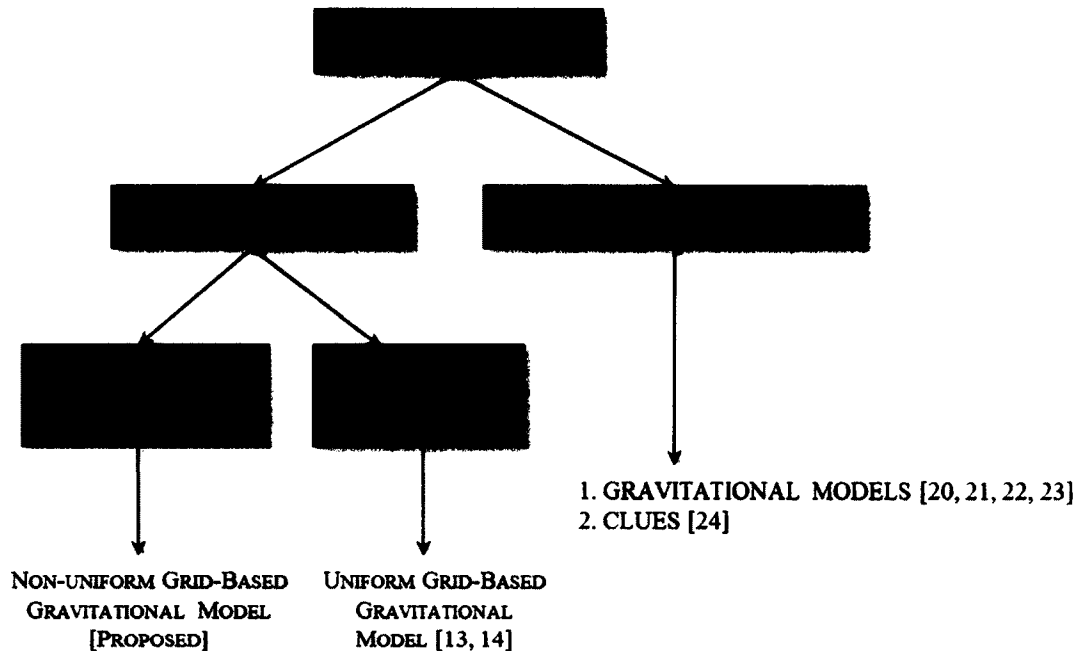


Figure 2.2: Data Shrinking Approaches

Grid-based data shrinking approaches employ grid-based partitioning to map the data in a grid structure. Initially, data is mapped on a grid structure and grid cell corresponding to each data point identified, and data points that are occupied in the same grid cells are also identified. Data points in the same grid cell move to other locations as a single unit. The movement of data points in each grid cell is then performed using the principle of data gravitation [13, 14]. Non-grid-based data shrinking approaches use the principle of data gravitation on individual data points. In these approaches, each data point is moved by a simulated movement of data points [20, 21, 22, 23, 24]. Grid-based approaches are faster, scalable, and computationally less expensive than non-grid/point-based approaches.

2.2.1 Point-Based Approach

In the past, many point-based data shrinking approaches have been used to employ the principle of data gravitation or gravitational transform [20, 21, 22, 23, 24]. The essence of all the approaches is as follows. Initially, a model of attraction (data gravitation) is assumed between the data points and a force of attraction is applied on a data point by its surrounding/neighbor data points. Then, this force of attraction enables the simulated movement of the data points. This process is applied for a specified number of iterations or until some stopping criterion is satisfied.

2.2.2 Grid-Based Approach

In the past only one grid-based data shrinking approach has been developed [13]. The overall process for this approach can be summarized as follows. Initially, multi-scale uniform grids are generated. Next, data points are mapped on the uniform grid structure and corresponding grid cells. Then, data points in each dense cell are moved toward the data centroid of the surrounding dense cells. This process is repeated until a specified movement threshold is achieved or for a specified number of iterations.

2.3 Feature Selection and Ranking

Feature selection is a process of identifying and selecting a subset of features from a given set of features to reduce the dimensionality of the data by optimizing an evaluation criterion. Feature selection reduces the dimensionality by removing irrelevant, noisy, and redundant features from the feature set [32, 33, 34, 35]. Application of feature selection as a preprocessing step in a data mining algorithm can greatly improve the accuracies and overall learning time of those algorithms. Feature selection techniques are essential and better techniques are always needed. Feature selection is frequently used in

data mining, especially in the fields of Bioinformatics, web mining, and other high dimensional data domains. The datasets in these domains may contain features that are irrelevant and unimportant and may have no predictive power. In fact, for some problems, only a small subset of features is usually relevant.

Feature selection techniques can be categorized into two categories, the filter model or the wrapper model [36, 37, 38, 39]. The filter model relies on general characteristics of the training data to select some features without involving any learning algorithm [40, 41, 42, 43, 44, 45]. The wrapper model requires one predetermined learning algorithm in the feature selection and uses its performance to evaluate and determine which features are selected. The wrapper methods tend to be more computationally expensive than the filter model. The filter methods are usually chosen due to its computational efficiency.

2.4 Classification

Classification is a supervised learning technique and many classification techniques have been developed [46]. However, the design of each classifier addresses a different issue, such as handling high dimensional and large datasets or improving the performance of the existing classifier. The common motivation that inspires scalable classifier design is the desire to develop a classifier capable of handling high dimensional and large datasets without significant loss in a performance parameter, such as speed or accuracy [47, 48, 49, 50, 51]. Handling high dimensional data in a data mining task, such as classification, is challenging because of the curse of dimensionality. Several methods have been developed to address the high dimensionality and large size of the dataset. The

SVM, KNN, and decision tree classifiers have been used extensively to design scalable classifiers [2, 46].

Decision tree based classification techniques, SLIQ and SPRINT are representative examples of scalable classifiers [49, 50]. The SLIQ algorithm consists of two phases, the tree growth phase and tree prune phase. It uses a one-time sort method instead of repeatedly sorting to split the numeric attribute. The algorithm is able to sort once rather than repeatedly, because it maintains separate lists for each attribute. It also maintains the 'class list' data structure that must remain in the memory all the time. It builds a single decision tree using the entire training dataset instead of using a sampled dataset. The size of the 'class-list' is the same as the number of data points; therefore, SLIQ can only handle data points that can be accommodated in the main memory. The SPRINT algorithm is an improvement over the SLIQ algorithm. The design goal of the researchers who developed SPRINT was to develop an accurate classifier for large datasets. SPRINT shares most of SLIQ's features, but it uses the 'attribute-list' instead of the 'class-list.' Unlike SLIQ, SPRINT has no memory restriction, and is fast and scalable [50].

A grid-based approach for the classification of network traffic data is presented in [19]. This method classifies data into normal and abnormal classes for anomaly detection. In this method, a two phase grid-based clustering algorithm was developed to partition the network traffic data. In the first phase, data points were divided into non overlapping cells for pre-clustering. In the second phase, k-hypercells clustering, the clusters returned from the algorithm were presented in the form of logical expressions to generate rules for the classification of network traffic data.

2.5 Clustering

Clustering is an unsupervised machine learning technique that groups the unlabeled data points into their natural groups within a given dataset. The driving principle of clustering is to have the data points in a cluster such that the data points within the clusters have high intra-cluster similarity and the data points between clusters have low inter-cluster similarity [2]. Clustering algorithms are commonly categorized in partitioning algorithms, hierarchical algorithms, density-based algorithms and grid-based algorithms [2, 52, 53, 54, 55, 56, 57]. They are also categorized in a specialized category called data shrinking based clustering algorithms [13, 14]. A detailed discussion about these clustering algorithms is as follows.

2.5.1 Partitioning-Based Clustering

Partitioning-based clustering algorithms employ an iterative approach to cluster the data points. This method starts with an initial configuration of k partitions. Initial k partitions are constructed by randomly or heuristically dividing the data points into k partitions specified by the user. Then, the data points in these k partitions are relocated or regrouped in other partitions by iteratively applying some relocation techniques. Well-known representative examples of partitioning-based clustering techniques are k-means, k-medoids, EM algorithm, fuzzy c-means, CLARA, CLARANS, and PAM [2].

2.5.2 Density-Based Clustering

Density-based clustering algorithms consider clusters as regions of high data point density separated by regions of low data points of density. Density-based clustering approaches start by growing a cluster until a density threshold is satisfied. A cluster that has a density greater or equal to the specified threshold is defined as a dense cluster and initially forms a cluster. Two dense clusters are merged if they share a common neighbor

[53, 54]. Well-known representative examples of density-based clustering techniques are DBSCAN, OPTICS, and DENCLUE [53, 54, 7].

2.5.3 Hierarchical Clustering

Hierarchical clustering algorithms create a tree-like decomposition of the given data [2]. Data is clustered at multiple levels of hierarchy. This method of clustering provides an opportunity to simultaneously analyze the clusters at different levels. Hierarchical clustering can start the clustering in bottom-up or top-down fashion. Hierarchical clustering techniques commonly use average-linkage, centroid-linkage, ward-linkage, single-linkage, and complete-linkage similarity criteria for clustering [2]. Dendrograms are generally used to represent the hierarchical decomposition of clusters. In most of the hierarchical clustering algorithms, once the merging of two clusters takes place, it cannot be undone. Therefore, most hierarchical clustering techniques are rigid. Well-known representative examples of hierarchical algorithms are CURE, CHAMELEON, ROCK, and BIRCH [52, 55, 56, 57]. Hierarchical clustering algorithms can be agglomerative or divisive.

2.5.3.1 Agglomerative Hierarchical Clustering

The agglomerative hierarchical clustering approaches perform clustering in bottom-up fashion. These approaches first assign each data point into its own cluster. Then, these single data points are merged with the other closest data points to form a bigger cluster using some similarity criterion. This process is repeated until all the data points are in one big cluster [2].

2.5.3.2 Divisive Hierarchical Clustering

The divisive hierarchical clustering approaches perform clustering in top-down fashion by assigning all the data points into one cluster. In the subsequent steps, these bigger clusters are split into smaller clusters. This process is repeated until all the data points are in one cluster or the desired number of clusters has been achieved [2].

2.5.4 Grid-Based Clustering

Grid-based clustering algorithms are based on grid-based localized learning. In these algorithms, a uniform or non-uniform grid structure is imposed on the data space, that is then partitioned into uniform or non-uniform grid cells. During this process, relevant statistical information is collected for each grid cell. Clustering is performed on grid cells instead of on individual data points. The most critical challenge of grid-based algorithms is the selection of the proper grid cell size. Finer grid cell sizes lead to the high computational cost and coarser grid cell sizes lead to poor clustering accuracies. Well-known representative examples of grid-based clustering algorithms are GRIDCLUS, DENCLUE, and WaveCluster [6, 7, 8]. Grid-based clustering algorithms are broadly categorized into uniform grid-based clustering and non-uniform grid-based clustering. These algorithms are discussed in the following sections. Figure 2.3 depicts grids used in clustering.

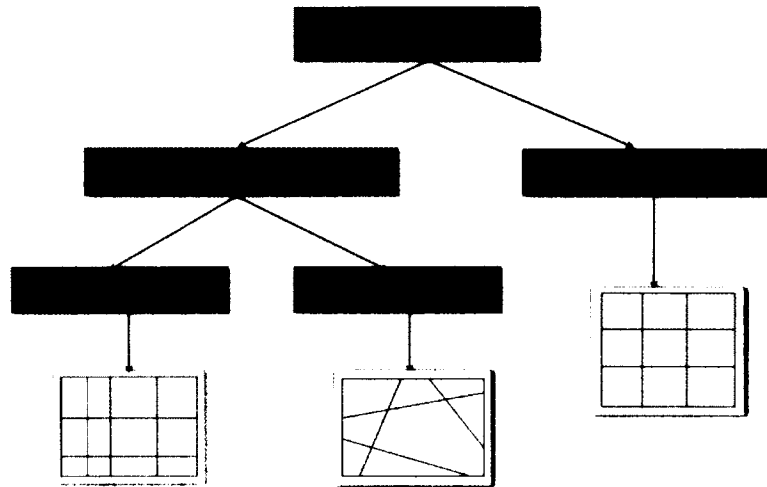


Figure 2.3: Types of Data Grids

2.5.4.1 Uniform Grid-Based Clustering

Uniform grid structures partition the data space using hyperplanes that are parallel to the axis. These grid structures are also called axis-parallel grid structure. It imposes the same size grid cells and do not take into account the underlying data distribution. Then, relevant statistical information is collected for each grid cell and clustering is performed. Well-known representative examples are WaveCluster, DENCLUE, and GRIDCLUS [8, 7, 6].

2.5.4.2 Non-Uniform Grid-Based Clustering

Non-Uniform grid-based clustering algorithms impose a data adaptive grid structure. Non-Uniform grid-based clustering algorithms offer significant performance improvement over other uniform grid-based clustering algorithms. Well-known representative examples are MAFIA, DESCRy, and MMNG [22, 11, 12].

2.5.5 Data Shrinking Based Clustering

A gravitational transform based clustering algorithm is presented in [20]. In this method, gravitational transform is applied to multi component image classification to

highlight the modes, or centers of high density regions, of data. The authors propose a simple model of attraction in which only mutual attraction of neighboring data points is enabled. This process is applied for a specified number of iterations. Finally, various clustering algorithms are applied to test the effectiveness of the proposed gravitational transform. Similarly, a new gravitational clustering algorithm that considers data points as an object in a gravitational field has been introduced [22]. In this algorithm, each data object is moved by simulating data movement for a specified number of iterations.

Finally, a cluster detection procedure is used to extract valid clusters at multiple levels of resolution. Following these methods, another gravitational clustering algorithm is presented in [23]. In this method, a force of attraction is applied between points, allowing each point to move slowly under the influence of the resultant force [23]. Data points that are close to each other during this movement process are merged to form a cluster. This merging process results in a hierarchical tree structure. Finally, clusters are obtained using an evaluation criterion. Further, a nonparametric clustering algorithm called CLUES is presented in [24]. It performs three functions: data shrinking, data clustering, and optimal cluster selection. The data shrinking process used in this algorithm is derived from the gravitational clustering. The movement of each data point is determined by the median of its k-nearest-neighbors because the median is more robust than the mean. The coordinates of each data point are updated in all iterations of the algorithm. This process is repeated until convergence is observed. Finally, data partitioning and optimal cluster selection is applied.

A multi-scale uniform grid-based data shrinking and clustering algorithm that simulates data movement toward the density gradient is presented in [13, 14]. This

technique is a three part method. First, data is mapped into grid cells. Then, data points in each dense cell move toward the data centroid of the surrounding cells. This process is repeated until a specified movement threshold is achieved for a specified number of iterations. Ultimately, clusters are detected at multiple scales, and cluster evaluation is performed to obtain the final clusters.

2.6 Conclusion

This chapter explores all related research paradigm in machine learning that are part of this dissertation. It starts by discussing the localized learning paradigm and then swiftly switches the discussion to the grid-based localized learning paradigm. It then explains and discusses the data shrinking, data shrinking techniques and related issues. Next, the clustering in general and research related to the dissertation such as grid-based clustering, hierarchical clustering, and data shrinking based clustering are discussed. Furthermore, it discusses related research in supervised machine learning paradigm.

CHAPTER 3

PRELIMINARIES OF GRID-BASED LOCALIZED LEARNING

Grid-based localized learning is a specialized form of learning in which data space is divided into small partitions called grid cells by imposing a grid structure. Thus, it is necessary to formally introduce frequently used terminology in this area. In this chapter, notations, formal definitions, and other important information relating to grid-based localized learning are provided.

The remainder of the chapter is organized as follows. In Section 3.1, basic notations used in explaining the algorithmic pseudo-code is discussed. In Section 3.2, formal definitions and theorems pertaining to grid-based localized learning are explained and discussed.

3.1 Notations

Let a set $\mathbb{X} = \{\mathcal{X}_i\}_{i=1}^N$ be a dataset of N d -dimensional data points, where $\mathbb{X} \subseteq \mathfrak{R}^d$ (\mathfrak{R} represents the set of real numbers), \mathcal{X}_i represents an element of \mathbb{X} . Let the element \mathcal{X}_i ($\mathcal{X}_i \in \mathbb{X}$) be a d -dimensional vector, which is represented by the vector $\vec{\mathcal{X}}_i = (\mathcal{X}_{i,1}, \dots, \dots, \mathcal{X}_{i,d})$. Let the set of d -dimensions be denoted by $\mathbb{D} = \{\mathcal{D}_j\}_{j=1}^d$. For $\forall j$, $1 \leq j \leq d$, let \mathcal{D}_j be normalized between $[0,1]$, where $[0,1] \subset \mathfrak{R}$. Let $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2 \times \dots \times \mathcal{D}_d$ be the d -dimensional data space in a unit hypercube $[0,1]^d \subset \mathfrak{R}^d$.

Then, let $\pi_j = [0,1]$ denote the value domain of the dimension \mathcal{D}_j , where $1 \leq j \leq d$. Let for $\forall \mathcal{D}_j$, $P_r = [l, h)$ denote a right-opened interval or partition and $P_c = [l, h]$ denote a closed interval or partition, where l denotes the lower bound and h denotes the upper bound of the partition. Let the value domain π_j of dimension \mathcal{D}_j be divided into \mathcal{K}^j mutually exclusive partitions. Let $I_{j,n} = [l_{j,n}, h_{j,n})$ be the n^{th} partition, where $1 \leq j \leq d$, $1 \leq n \leq \mathcal{K}^j$. Let $I_j = \{I_{j,1}, \dots, I_{j,\mathcal{K}^j}\} = \{[l_{j,1}, h_{j,1}), \dots, [l_{j,\mathcal{K}^j}, h_{j,\mathcal{K}^j})\}$ be the total-ordered set (I_j, \leq) that denotes the partitions in dimension \mathcal{D}_j such that $(l_{j,1} < l_{j,2} < \dots < l_{j,\mathcal{K}^j})$ $(h_{j,1} < h_{j,2} < \dots < h_{j,\mathcal{K}^j})$. Let Object_i be the total number of data points in the grid cell C_i . Let Volume_i be the volume of the grid cell C_i . Let ρ_i be the density of the grid cell C_i . Let L_i be the length of the i^{th} partition of the grid cell C_i .

3.2 Formal Definitions

Using the above notations, the formal definitions of relevant terminologies in grid-based localized learning are presented here.

Definition 3.1 (Grid): A grid G on a d -dimensional unit hypercube data space \mathcal{D} that partitions the data space into $\prod_{j=1}^{j=d} \mathcal{K}^j$ number of partitions is given by a d -ary Cartesian product over d totally-ordered sets $I_1, I_2, \dots, \dots, I_d$. A d -dimensional grid G is given by Equation 3.1 or 3.2:

$$G = I_1 \times I_2 \times \dots \times I_d, \quad \text{Eq. 3.1}$$

$$G = \left\{ \left(I_{1,n_1}, I_{2,n_2}, \dots, I_{j,n_j}, \dots, I_{d,n_d} \right) \mid I_{j,n_j} \in I_j \right\}. \quad \text{Eq. 3.2}$$

Definition 3.2 (Uniform Grid): A uniform/fixed size grid G_{UNIFORM} on a d -dimensional data space that partitions the data space \mathcal{D} into $\prod_{j=1}^{j=d} \mathcal{K}^j$ number of partitions is a d -ary

Cartesian product over d totally-ordered sets I_1, I_2, \dots, I_d such that $I_1 = I_2 = \dots = I_d = I$, $\mathcal{K}^j = \mathcal{K}$ for $\forall \mathcal{D}_j$ and $|h_{j,n_j} - l_{j,n_j}| = 1/\mathcal{K}$, for $\forall n_j$. A d -dimensional uniform grid $G_{UNIFORM}$ is given by Equation 3.3 or 3.4. Figure 3.1 depicts a two-dimensional uniform grid:

$$G_{UNIFORM} = I_1 \times I_2 \times \dots \times I_d, \quad \text{Eq. 3.3}$$

$$G_{UNIFORM} = \left\{ (I_{1,n_1}, I_{2,n_2}, \dots, I_{j,n_j}, \dots, I_{d,n_d}) \mid I_{j,n_j} \in I_j \right\}. \quad \text{Eq. 3.4}$$

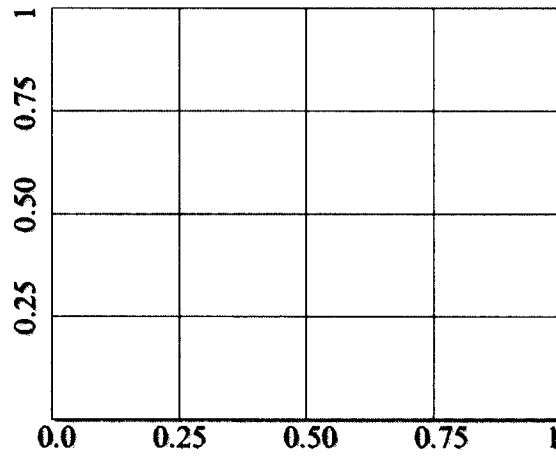


Figure 3.1: A Two-Dimensional Uniform Grid

Definition 3.3 (Non-Uniform Grid): A non-uniform grid $G_{ADAPTIVE}$ on a d -dimensional data space that partitions the data space \mathcal{D} into $\prod_{j=1}^{j=d} \mathcal{K}^j$ number of partitions is a d -ary Cartesian product over d totally ordered sets I_1, I_2, \dots, I_d such that $I_1 \neq I_2 \neq \dots \neq I_d$ and $I_1 \neq I_2 \neq \dots \neq I_d$. A d -dimensional data adaptive grid $G_{ADAPTIVE}$ is given by Equation 3.5 or 3.6. Figure 3.2 depicts a two-dimensional non-uniform grid:

$$G_{ADAPTIVE} = I_1 \times I_2 \times \dots \times I_d, \quad \text{Eq. 3.5}$$

$$G_{ADAPTIVE} = \left\{ (I_{1,n_1}, I_{2,n_2}, \dots, I_{j,n_j}, \dots, I_{d,n_d}) \mid I_{j,n_j} \in I_j \right\}. \quad \text{Eq. 3.6}$$

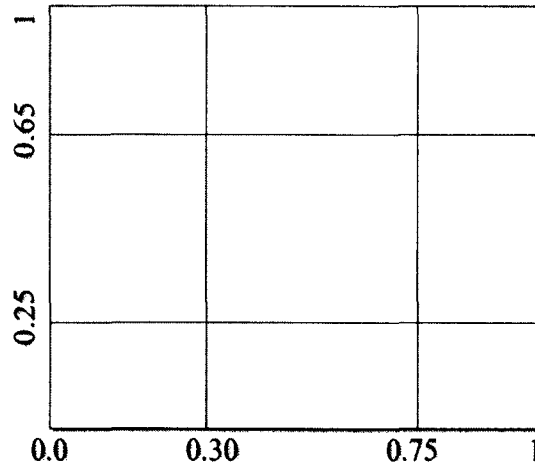


Figure 3.2: A Two-Dimensional Non-Uniform Grid

Definition 3.4 (Grid Cell): A grid cell C in a d -dimensional grid G is a d -tuple such that each element l_{j,n_j} of the d -tuple represents a partition $[l_{j,n_j}, h_{j,n_j})$ in a dimension. A d -dimensional grid cell C is given by Equation 3.7 or 3.8 and is depicted in Figure 3.3:

$$C = (l_{1,n_1}, l_{2,n_2}, \dots, l_{j,n_j}, \dots, l_{d,n_d}), \quad \text{Eq. 3.7}$$

$$C = ([l_{1,n_1}, h_{1,n_1}), \dots, [l_{j,n_j}, h_{j,n_j}), \dots, [l_{j,n_d}, h_{j,n_d})). \quad \text{Eq. 3.8}$$

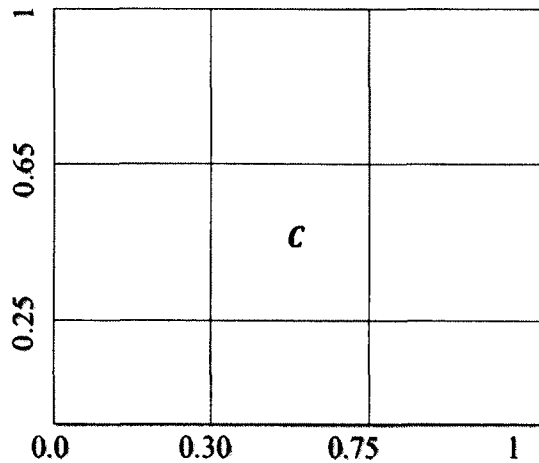


Figure 3.3: A Grid Cell Representation

Definition 3.5 (Uniform Grid Cell): A grid cell $C_{Uniform}$ in a d -dimensional uniform grid G is a d -tuple such that each element l_{j,n_j} , of the d -tuple represents a partition $[l_{j,n_j}, h_{j,n_j})$ in a dimension where $|h_{j,n_j} - l_{j,n_j}| = 1/\mathcal{K}$, for $\forall n_j$. A uniform grid cell $C_{Uniform}$ is given by Equation 3.9 or 3.10:

$$C_{Uniform} = (l_{1,n_1}, l_{2,n_2}, \dots, l_{j,n_j}, \dots, l_{d,n_d}), \quad \text{Eq. 3.9}$$

$$C_{Uniform} = ([l_{1,n_1}, h_{1,n_1}), \dots, [l_{j,n_j}, h_{j,n_j}), \dots, [l_{j,n_d}, h_{j,n_d})). \quad \text{Eq. 3.10}$$

Definition 3.6 (Non-Uniform Grid Cell): A grid cell $C_{Non-uniform}$ in a d -dimensional non-uniform grid G is a d -tuple such that each element l_{j,n_j} , of the d -tuple represents a partition $[l_{j,n_j}, h_{j,n_j})$ in a dimension where $|h_{j,n_j} - l_{j,n_j}| \neq 1/\mathcal{K}$, for $\forall n_j$. A d -dimensional grid cell $C_{Non-uniform}$ is given by Equation 3.11 or 3.12:

$$C_{Non-uniform} = (l_{1,n_1}, l_{2,n_2}, \dots, l_{j,n_j}, \dots, l_{d,n_d}), \quad \text{Eq. 3.11}$$

$$C_{Non-uniform} = ([l_{1,n_1}, h_{1,n_1}), \dots, [l_{j,n_j}, h_{j,n_j}), \dots, [l_{j,n_d}, h_{j,n_d})). \quad \text{Eq. 3.12}$$

Definition 3.7 (Empty Grid Cell): A grid cell C in a d -dimensional grid G is called an empty grid cell if, and only if, no data point $\vec{x}_i = (x_{i,1}, \dots, x_{i,j}, \dots, x_{i,d})$ exists such that $l_{j,n_j} \leq x_{i,j} < h_{j,n_j}$ for $\forall x_{i,j}$. A d -dimensional empty grid cell C is given by Equation 3.13:

$$C = ([l_{1,n_1}, h_{1,n_1}), \dots, [l_{j,n_j}, h_{j,n_j}), \dots, [l_{j,n_d}, h_{j,n_d})) = \emptyset. \quad \text{Eq. 3.13}$$

Definition 3.8 (Non-Empty Grid Cell): A grid cell C in a d -dimensional grid G is called a non-empty grid cell if, and only if, at least one data point $\vec{x}_i = (x_{i,1}, \dots, x_{i,j}, \dots, x_{i,d})$

exists such that $l_{j,n_j} \leq \mathcal{X}_{i,j} < h_{j,n_j}$ for $\forall \mathcal{X}_{i,j}$. A d -dimensional non-empty grid cell C is given by Equation 3.14:

$$C = \left([l_{1,n_1}, h_{1,n_1}), \dots, [l_{j,n_j}, h_{j,n_j}), \dots, [l_{j,n_d}, h_{j,n_d}) \right) \neq \emptyset. \quad \text{Eq. 3.14}$$

Definition 3.9 (Neighboring/Connected Grid Cell): Let C_p and C_q be two grid cells in a d -dimensional grid G . Let C_p and C_q represent d -tuple $C_p = (l_{1,p_1}, \dots, l_{j,p_j}, \dots, l_{d,p_d})$ and $C_q = (l_{1,q_1}, \dots, l_{j,q_j}, \dots, l_{d,q_d})$, respectively. Grid cells C_p and C_q are called neighboring/connected grid cells if, and only if, $|l_{j,p_j} - l_{j,q_j}| \leq 1$ for $\forall (1 \leq j \leq d)$.

Definition 3.10 (Non-Empty Neighboring Grid Cell): Let two d -dimensional grid cells, C_p and C_q , be given by $C_p = (l_{1,p_1}, \dots, l_{j,p_j}, \dots, l_{d,p_d})$ and $C_q = (l_{1,q_1}, \dots, l_{j,q_j}, \dots, l_{d,q_d})$, respectively. Grid cells C_p and C_q are called non-empty neighboring grid cells if, and only if, $C_p \neq \emptyset$, $C_q \neq \emptyset$ and $|l_{j,p_j} - l_{j,q_j}| \leq 1$ for $\forall (1 \leq j \leq d)$.

Definition 3.11 (Grid Cell Volume): Let grid cell C_i be a d -tuple in a d -dimensional grid G such that each element l_{j,n_j} of the d -tuple represents a partition $[l_{j,n_j}, h_{j,n_j})$ in a dimension. Let L_i be the length of the i^{th} partition in the d -tuple. The volume $Volume_i$ of a grid cell C_i is given by Equation 3.15:

$$Volume_i = \frac{1}{(L_1 \times \dots \times L_i \times \dots \times L_d)}. \quad \text{Eq. 3.15}$$

Definition 3.12 (Grid Cell Density): Let C_i be a grid cell in a d -dimensional grid G , let $Object_i$ be the total number of data points in the grid cell C_i , let $Volume_i$ be the total volume of the grid cell C_i and let ρ_i be the density of the grid cell C_i . The density ρ_i of a

grid cell C_i is given by the ratio of $Object_i$ and $Volume_i$. It is expressed by

Equation 3.16:

$$\rho_i = \frac{Object_i}{Volume_i}. \quad \text{Eq. 3.16}$$

Definition 3.13 (Dense Grid Cell): Let C_i be a grid cell in a d -dimensional grid G , let ρ_i be its density, and let Th_ρ be a density threshold. Grid cell C_i is called a dense grid cell if, and only if, the density ρ_i is greater than or equal to Th_ρ . It is expressed by

Equation 3.17:

$$C_i = \begin{cases} \text{Sparse, if } \rho_i < Th_\rho \\ \text{Dense, if } \rho_i \geq Th_\rho \end{cases} \quad \text{Eq. 3.17}$$

Definition 3.14 (r^{th} Rank Neighbor): Let grid cell C and C_p be represented by d -tuples $(I_{1,n_1}, \dots, I_{d,n_d})$ and $(I_{1,p_1}, \dots, I_{d,p_d})$, respectively. Grid cell C_p is called the r^{th} rank neighbor of the grid cell C if, and only if the following condition is satisfied. This condition is expressed in Equation 3.18:

$$I_{j,p_j} = \begin{cases} I_{j,n_j} + 1 \text{ or } I_{j,n_j} - 1, & \forall j, (1 \leq j \leq r) \\ I_{j,n_j}, & \forall j, (r + 1 \leq j \leq d) \end{cases} \quad \text{Eq. 3.18}$$

Definition 3.15 (Data Centroid): Let C_i be a grid cell that contains a set \mathbb{X}_i of k data points $\mathbb{X}_i = \{\vec{x}_{i1}, \dots, \vec{x}_{ik}\}$, where $\mathbb{X}_i \subset \mathbb{X}$. The data centroid \vec{c}_i of the grid cell C_i is given by Equation 3.19:

$$\vec{c}_i = \frac{1}{k} (\sum_1^k \vec{x}_{ik}). \quad \text{Eq. 3.19}$$

Definition 3.16 (Overlapping-Cell): Let C_i be a grid cell that contains a set \mathbb{X}_i of k data points $\mathbb{X}_i = \{\vec{x}_{i1}, \dots, \vec{x}_{ik}\}$, where $\mathbb{X}_i \subset \mathbb{X}$. The grid cell C_i is called an overlapping-cell if it contains training samples from multiple classes.

Definition 3.17 (Non-Overlapping Cell): Let C_i be a grid cell that contains a set \mathbb{X}_i of k data points $\mathbb{X}_i = \{\vec{\mathcal{X}}_{i1}, \dots, \dots, \vec{\mathcal{X}}_{ik}\}$, where $\mathbb{X}_i \subset \mathbb{X}$. The grid cell C_i is called a non-overlapping cell if it only contains the training samples of a single class.

Definition 3.18 (Micro-Partition): Let m_r be a micro-partition that contains k data points $(m_{r,1}, m_{r,u}, \dots, \dots, m_{r,k})$. A micro-partition m_r is a smallest non-overlapping unit of data points in which data points are in close proximity ($|m_{r,u} - m_{r,(u+1)}| \approx \varepsilon$, where ε is a small number) with each other.

Definition 3.19 (Average Linkage): Let m_r , and m_{r+1} be two contiguous micro-partitions that are given by sets $m_r = (m_{r,1}, \dots, m_{r,k})$ and $m_{r+1} = (m_{r+1,1}, \dots, m_{r+1,k})$, respectively. The average linkage between two contiguous micro-partitions is defined by Equation 3.20:

$$AVERAGE(m_r, m_{r+1}) = \frac{1}{(k \cdot k)} \sum_{i=1}^k \sum_{j=1}^{k'} |m_{r,i} - m_{r+1,j}|. \quad \text{Eq. 3.20}$$

Definition 3.20 (Centroid Linkage): Let m_r , and m_{r+1} be two contiguous micro-partitions in the transformed space that are given by sets $m_r = (m_{r,1}, \dots, m_{r,k})$ and $m_{r+1} = (m_{r+1,1}, \dots, m_{r+1,k})$, respectively. The centroid linkage between two contiguous micro-partitions is given by Equation 3.21:

$$CENTROID(m_r, m_{r+1}) = |\overline{m_r} - \overline{m_{r+1}}|, \quad \text{Eq. 3.21}$$

where $\overline{m_r} = \frac{1}{k} \sum_{i=1}^k m_{r,i}$, and $\overline{m_{r+1}} = \frac{1}{k'} \sum_{j=1}^{k'} m_{(r+1),j}$.

Definition 3.21 (Ward Linkage): Let m_r , and m_{r+1} be two contiguous micro-partitions in the transformed space that are given by sets $m_r = (m_{r,1}, \dots, m_{r,k})$ and $m_{r+1} = (m_{r+1,1}, \dots, m_{r+1,k})$, respectively. The ward linkage between two contiguous micro-partitions is given by Equation 3.22:

$$WARD(m_r, m_{r+1}) = (k * k') \frac{|\overline{m_r} - \overline{m_{r+1}}|}{(k+k')}, \quad \text{Eq. 3.22}$$

where $\overline{m_r} = \frac{1}{k} \sum_{i=1}^k m_{r,i}$, and $\overline{m_{r+1}} = \frac{1}{k'} \sum_{j=1}^{k'} m_{(r+1),j}$.

Definition 3.22 (Z-Score Normalization): Assume A is a numeric attribute, its mean is μ_A , its variance is σ_A , and a specific attribute value is $Value_A$. Attribute value $Value_A$ is mapped to a new attribute value $Value'_A$ by computing the following equation:

$$Value'_A = \frac{(Value_A - \mu_A)}{\sigma_A}. \quad \text{Eq. 3.23}$$

Definition 3.23 (Min-Max Normalization): Min-max normalization performs a linear transformation on the attribute values. Assume A is a numeric attribute, its maximum value is Max_A , its minimum value is Min_A , and a specific attribute value is $Value_A$. Attribute value $Value_A$ is mapped to a new attribute value $Value'_A$ in the range of $[NewMin_A, NewMax_A]$ by computing the following equation:

$$Value'_A = \frac{(Value_A - Min_A) \times (NewMax_A - NewMin_A)}{(Max_A - Min_A)} + NewMin_A. \quad \text{Eq. 3.24}$$

Theorem 1: Grid-Based Neighborhood

Let G be a grid on a d -dimensional data space \mathcal{D} that partitions the data space into mutually exclusive intervals or partitions. Let C_u be a d -dimensional grid cell that is a d -tuple $C_u = (I_{1,n_1}, I_{2,n_2}, \dots, I_{j,n_j}, \dots, I_{d,n_d})$ such that each element of the tuple represents a partition in the corresponding dimension. Then, a d -dimensional grid cell C_u can have $C_{Neighbor}$ distinct neighboring grid cells that are given by Equation 3.25, where S_j is the number of changes in the partition index value I_{j,n_j} in dimension \mathcal{D}_j that satisfies the neighborhood criteria:

$$C_{Neighbor} = \prod_{j=1}^d S_j - 1. \quad \text{Eq. 3.25}$$

Proof: Let a d-tuple $(I_{1,p_1}, \dots, I_{j,p_j}, \dots, I_{d,p_d})$ represent a grid cell C_p . The grid cell C_p is the neighboring grid cell of cell $C_u = (I_{1,n_1}, \dots, I_{j,n_j}, \dots, I_{d,n_d})$ if, and only if, $I_{j,p_j} = \{ I_{j,n_j} - 1 \text{ or } I_{j,n_j} + 1 \text{ or } I_{j,n_j}, \forall j, (1 \leq j \leq d)$. Therefore, each element I_{j,p_j} of a neighboring grid cell C_p can have a maximum of three values that satisfy the neighborhood criterion. If S_j represents all possible changes for dimension \mathcal{D}_j , then the number of neighboring grid cells is given by Equation 3.26:

$$C_{Neighbor} = (S_1 * \dots * S_j * \dots * S_d) - 1, \quad \text{Eq. 3.26}$$

$$C_{Neighbor} = (S_1 * \dots * S_j * \dots * S_d) - 1 = \prod_{j=1}^d S_j - 1. \quad \text{Eq. 3.27}$$

It should be noted that -1 in Equation 3.26 indicates $I_{j,p_j} = I_{j,n_j} \forall j, (1 \leq j \leq d)$

when $C_p = C_u$. Equation 3.26 can also be represented in the form of Equation 3.27.

CHAPTER 4

GRID-BASED LOCALIZED LEARNING FOR DATA PREPROCESSING

Most real world data is low quality, and the data used for the data mining tasks may be incomplete, noisy, inconsistent, and sparse. Consequently, it is necessary to improve the quality of the data by addressing these data deficiencies prior to data analysis through a series of steps collectively called data preprocessing. There are several challenges in preparing this data for data mining tasks such as clustering and classification among others. These challenges are categorized into challenges related to the characteristics of the raw data such as noisy, missing, and inconsistent data values and into challenges related to the characteristics of the data such as sparseness and the curse of dimensionality in multidimensional data space.

Both these sets of challenges severely affect the data analysis and may lead to low quality and misleading conclusions. Therefore, data preprocessing is necessary before performing any type of data mining tasks. Many techniques have been developed to handle the noise, incomplete and inconsistent data. Similarly, many techniques have been developed to mitigate the effect of the curse of dimensionality and the sparseness of the data. The sparseness of the data, which is caused by the curse of the dimensionality, severely undermines the performance of data mining algorithms. Because of this potential deterioration of the performance, one emphasis of the research presented in this

dissertation is to develop better sparseness reduction algorithms and frameworks and integrate them with the clustering algorithms.

The remainder of this chapter is organized as follows. In Section 4.1, a brief explanation of various data preprocessing techniques is provided. In Section 4.2, a discussion about data sparseness, its detrimental effects and sparseness reduction techniques are provided. In Section 4.3, research motivation for the non-uniform grid-based sparseness reduction technique is discussed. In Section 4.4, an experimental study is presented to demonstrate the advantages of the non-uniform grid-based sparseness reduction technique. In Section 4.5, the conclusions of this chapter are presented.

4.1 Data Preprocessing

Data preprocessing refers to the process of improving the quality of data for the ease of the data mining or knowledge discovery process. Data preprocessing is a collection of a wide variety of operations. The process includes data cleaning operations, which usually compose the first set of operations performed on the data. The second set of operations is called data transformation operations, which converts the data into a specified format. The third set of operations is referred to as data reduction operations, which includes operations to reduce data such as aggregation and dimensionality reduction. The fourth set of operations is referred as data shrinking operations. It includes operations regarding sparseness reduction. Data processing can improve the overall quality of data and the data mining tasks for knowledge discovery [2]. A brief discussion about all four sets of operations is given below.

- 1. Data Cleaning:** Data cleaning refers to the set of operations performed to clean the data by removing noise from the data, filling in missing data values, and resolving

inconsistent data values. Common noise removal operations are binning, regression, and clustering. Common operations for filling in missing values involve the use of a global constant, the use of an attribute mean, and the use of a most probable value.

Common operations for resolving inconsistent values are the use of domain knowledge and the use of rules discovery to find inconsistent relationships [2].

- 2. Data Transformation:** Data transformation refers to the set of operations that transform the data into representations which are appropriate for the data mining task at hand [2]. The set of data transformation operations consists of data smoothing, aggregation, generalization, normalization, and attribute construction. Data smoothing involves binning, regression, and clustering. Data aggregation involves data summarization. Data generalization involves replacing raw data by higher level concepts. Data normalization involves scaling data values into the specified range. Attribute construction involves extracting new attributes from the given set of attributes.
- 3. Data Reduction:** Data reduction refers to the set of operations that are applied to obtain a reduced representation of the data without seriously compromising the integrity of the original data [2]. Data reduction operations consist of data aggregation, attribute subset selection, dimensionality reduction, and sample reduction. Data aggregation involves data summarization. Attribute subset selection involves removing irrelevant, weak, or redundant attributes. Dimensionality reduction involves reducing dimensions by applying wavelet transform, principal component analysis, and Fourier transform, among other methods. Sample reduction involves the use of histograms, clustering, parametric models, and sampling techniques [2].

4. Data Shrinking: Data shrinking refers to the process of sparseness reduction through the simulated movement of data points using the principle of data gravitation. In the simulated movement of data points, data points are attracted by their surrounding neighborhood because of data gravitation, and they move along the direction of the density gradient [13, 14, 20, 21, 22, 23, 24]. Data shrinking techniques include grid-based approaches and point-based approaches.

4.2 Data Sparseness

Sparseness of the data refers to thinly scattered data points in the feature space. Sparseness is a common characteristic of multidimensional data. In sparse data, natural groups, or clusters of data points, are not well separated or well demarcated and have blurry cluster boundaries. The sparseness of the data increases as the dimensions increase because the number of data points required for filling the data space also increases exponentially. Therefore, data points are thinly scattered and lack distinct cluster boundaries, and the capability of clustering algorithms to detect clusters accurately is adversely affected in these datasets [13, 14]. Thus, it is necessary to develop sparseness reduction techniques that can override the sparseness of multidimensional data effectively.

Furthermore, the sparseness of multidimensional data is usually handled by a specialized data preprocessing strategy called data movement or data shrinking [13, 14, 20, 21, 22, 23, 24]. These data movement algorithms reduce the sparseness of multidimensional data while maintaining the original dimensional space. Data movement approaches diminish the sparseness of multidimensional data by moving data points along the direction of the density gradient, thus, providing more condensed and

demarcated clusters in the original dimensional space while retaining the dimensions [13, 14]. These data movement algorithms are iterative and require a specified number of iterations or stopping criteria. Existing sparseness reduction techniques is either point-based approaches or grid-based approaches [20, 21, 22, 23, 24, 13, 14].

4.3 Research Motivation

The existing grid-based data shrinking algorithms use uniform grid structure [13, 14]. However, the uniform grid structure is insensitive to underlying data distribution and does not project the underlying distribution of the data. Consequently, the uniform grid structure does not shrink all data points effectively. This problem is further aggravated as the number of dimensions increases. Existing sparseness reduction approaches are either inherently unstable or time consuming. Non-Uniform/adaptive grid structure is data driven and captures the underlying data distribution in every dimension. Grid-based approaches are fast, scalable and require less iteration than point-based approaches [20, 21, 22, 23, 24]. Therefore, an experimental study is conducted on synthetic and real multidimensional datasets to evaluate and demonstrate the effectiveness of the adaptive grid-based data shrinking approach.

4.3.1 Limitations of Existing Techniques

The limitations of existing data shrinking techniques are the instability of the shrinking and imposition of the uniform grid structure. These limitations are discussed below.

1. Sensitivity towards the order of Input Data Points: In the existing algorithms, there is no order specified in processing the data points [13, 14, 20, 21, 22, 23, 24].

The order in which data points are moved to other positions depends on the order in

which data points are stored. Thus, if the order in which the data points are given as input to the algorithm changes, the order in which the data points are moved will also change. This change in order then changes the final output of the data shrinking, giving the existing data shrinking algorithms inherent sensitivity towards the order of the input data points.

2. **Imposition of Uniform Grid Structure:** In the existing grid-based data shrinking algorithm, a sequence of uniform grid sizes is imposed on all dimensions [13, 14]. The algorithm imposes a global grid cell size on all dimensions and ignores the unique underlying data distribution in individual dimensions.

4.3.2 Advantages of Non-Uniform Grid

In grid-based clustering approaches both uniform and non-uniform grids are used. non-uniform/adaptive grids offer various advantages over uniform/fixed size grids. These advantages are explained below.

1. **Splitting Dimensions in Low Density Regions:** In a grid-based algorithm, dimensions are partitioned through split points; each point then becomes a cutting plane for multidimensional data. A cutting plane must partition a dimension in a low density region and discriminate clusters as much as possible [26]. Adaptive partitions are based on the data distribution in a dimension and split dimensions at low density regions [26].
2. **Computational Efficiency:** There are fewer nonempty grid cells for a specified number of partitions in every dimension than nonempty grid cells in a uniform grid. The fewer nonempty cells reduce the overall computational time for a non-uniform grid-based algorithm [26].

4.4 Experimental Study

In this section, an experimental study is presented. These experiments are conducted to demonstrate the effect of sparseness with increasing dimensions and the advantages of a non-uniform adaptive grid over uniform grid.

4.4.1 Datasets

Both real and synthetic datasets are used for experiments and to compare the uniform partitioning and non-uniform partitioning. A detailed description of each of these datasets is as follows:

- 1. Wine Recognition Dataset:** The real dataset that is used in these experiments is the Wine Recognition dataset. The Wine Recognition dataset is used for the comparative study of uniform and non-uniform grid-based shrinking. This dataset has 13 dimensions and 178 data points. The dataset contains three clusters, and each cluster contains 59, 71, and 48, respectively. The dataset is available at the UCI machine learning archive [58].
- 2. Synthetic Dataset:** For these experiments, a set of synthetic datasets is generated with dimensions ranging from 5 to 60 with increments of five dimensions, and every dataset has 10,000 data points. The size of dataset is kept constant because this synthetic dataset is used to demonstrate the effect of sparseness with increasing dimensions while keeping the dataset size constant. Each dataset contains two clusters, each of which has an equal number of data points in respective datasets. Both the clusters are generated from a normal distribution with means 10, -10 and a standard deviation of 3.

4.4.2 Effect of Sparseness

Multidimensional datasets are sparse and the sparseness of multidimensional data increases as dimensions increase. An experimental study is conducted on synthetic datasets to demonstrate this effect and it is presented in Figure 4.1.

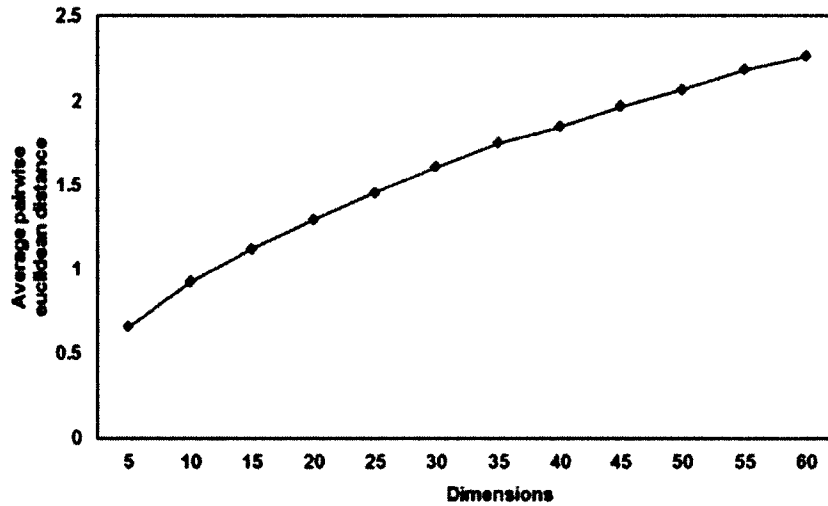


Figure 4.1: Average Pairwise Euclidean Distance v/s Dimensions

The sparseness of the datasets is computed by calculating the average pairwise Euclidean distance between data points. Figure 4.1 shows the average pairwise Euclidean distance, which is plotted as a function of increasing dimensions. The average pairwise Euclidean distance is given below in Equation 4.1:

$$\text{Average Pairwise Distance} = \frac{\sum_{k=1}^n \sum_{i=1}^n \sqrt{\sum_{j=1}^d (x_{i,j} - x_{k,j})^2}}{n^2}. \quad \text{Eq. 4.1}$$

It is demonstrated from the plot that data sparseness increases with increasing dimensions for a constant number of data points. Similarly, the increase in the number of data points would result in the same exponential characteristic but the rate of increase in the distance between the data points would be less as compared to the dataset with less number of data points.

4.4.3 Comparative Study

In this comparative study, two sets of experiments are conducted. The first set of experiments is conducted to compare the uniform and non-uniform grid-based partitioning. The second set of experiments is conducted to compare the uniform and non-uniform grid-based shrinking. These studies are discussed below.

4.4.3.1 Comparison of Partitioning Methods

A comparative study is performed on a synthetic dataset to demonstrate the advantage of non-uniform grid-based partitioning over the uniform grid-based partitioning. In this study, a comparison of the total number of nonempty grid cells that occupy data points is performed between uniform and non-uniform grids for the given synthetic datasets. Uniform grid partitions are generated using the algorithm presented in and non-uniform grid partitions are generated using the non-uniform grid generation presented in Chapter 6. Plots of the comparative study are presented below. Three cases are considered to compare the two partitioning methods.

Case 1: In this experiment, uniform and non-uniform grid generation algorithms are applied, and two uniform partitions and two non-uniform partitions are generated for each dimension. Figure 4.2 presents a comparison of the log of the ratio of the total grid cells and the total non-empty grid cells occupied by all the data points in both approaches. It can be inferred from the plot that, in adaptive grid-based partitioning, data points are occupied in fewer grid cells in almost all cases, as compared to the uniform grid-based partitioning.

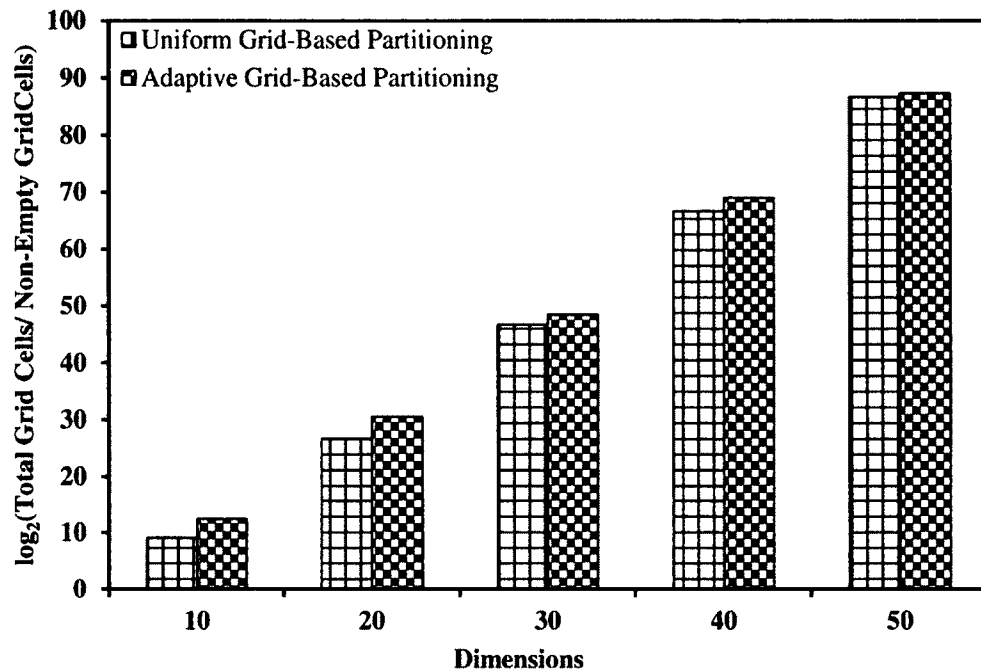


Figure 4.2: \log_2 (Total Grid Cells/Non-Empty Grid Cells) v/s Dimensions

Case 2: In this example, a uniform grid generation algorithm is applied, and three uniform and non-uniform partitions are generated for each dimension. Figure 4.3 depicts a comparison of the log of the ratio of total grid cells and total non-empty grid cells occupied by all the data points in both approaches. It can be inferred from the plot that in an adaptive grid-based partitioning data points occupy fewer grid cells than the uniform grid-based partitioning.

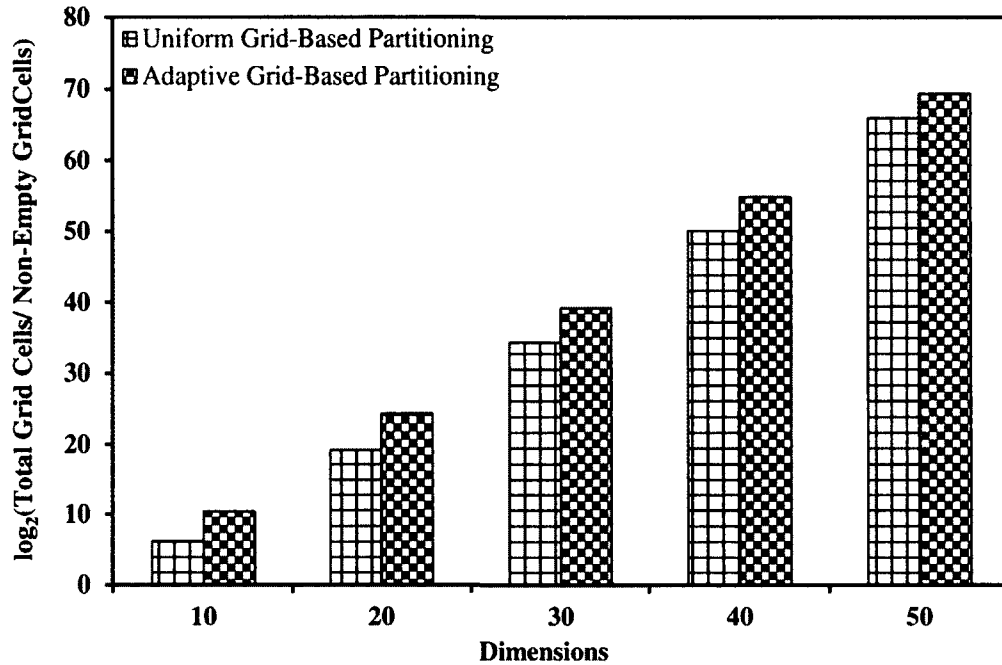


Figure 4.3: \log_2 (Total Grid Cells / Non-Empty Grid Cells) v/s Dimensions

Case 3: In this example, both uniform and non-uniform grid generation algorithms are applied, and four uniform partitions and four non-uniform partitions are generated for each dimension. Figure 4.4 depicts a comparison of the log of the ratio of the total grid cells and the total non-empty grid cells occupied by all the data points in both approaches. It can be inferred from the figure that in an adaptive grid-based partitioning data points are occupied in fewer grid cells or an equal number of grid cells than the uniform grid-based partitioning. It also implies that non-uniform grid-based shrinking and clustering algorithms can be computationally less expensive or may incur the same computational cost.

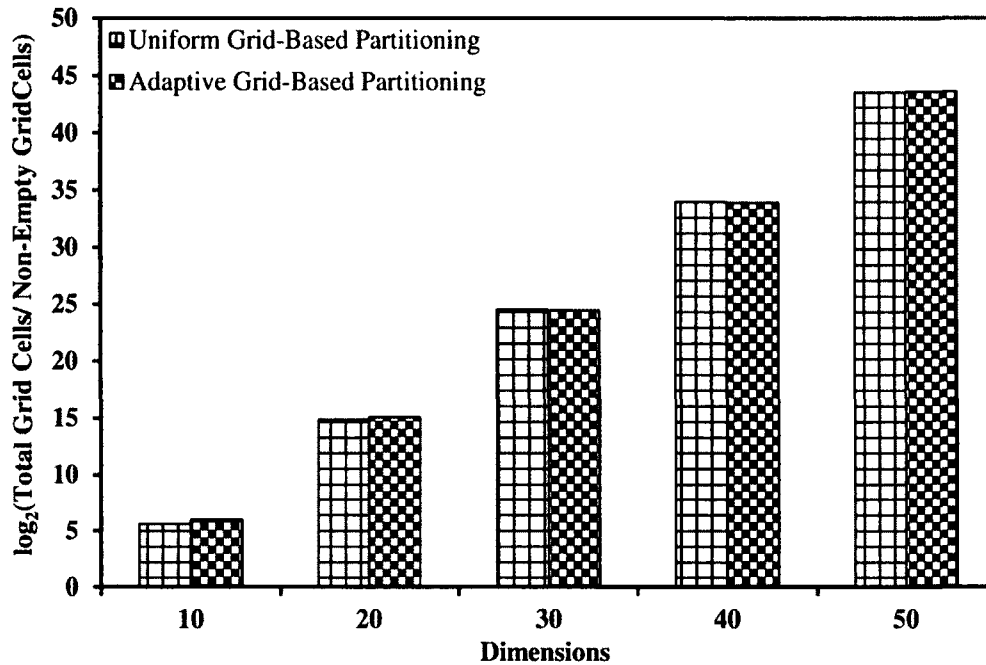


Figure 4.4: \log_2 (Total Grid Cells/Non-Empty Grid Cells) v/s Dimensions

4.4.3.2 Comparison of Shrinking Methods

To compare the uniform and non-uniform grid-based shrinking algorithms, these algorithms are applied on the Wine Recognition dataset. Uniform grid-based shrinking is performed using an existing data shrinking algorithm presented in [13, 14]. Similarly, non-uniform grid-based shrinking is performed using the non-uniform grid-based shrinking algorithm presented in Chapter 6. Both uniform and non-uniform grid-based shrinking algorithms are applied to the grid structure with three partitions for each dimension. These methods are compared based on the energy, wavelet entropy, and information entropy of the data in principal component space. Principal components are obtained on the Wine Recognition dataset in three conditions. These conditions are, after uniform grid-based shrinking, after non-uniform grid-based shrinking, and without shrinking. Plots of the comparative study are presented below. In this experimental study wavelet

entropy, energy, and information entropy are computed for each dimension in principal component space. Case 1, Case 2, and Case 3 below presents these experiments.

Case 1: In this experimental study, wavelet entropy is computed corresponding to each dimension in principal component space, and the percentage of the wavelet entropy contributed by each dimension is obtained. Finally, a plot is obtained that depicts the cumulative percentage of the wavelet entropy for each set of dimension in principal component space. Figure 4.5 depicts a comparison of the cumulative wavelet entropy. It can be observed from the plot that after non-uniform grid-based shrinking principal components retain the lowest cumulative wavelet entropy, which indicates that after performing non-uniform grid-based shrinking each dimension has less disorder.

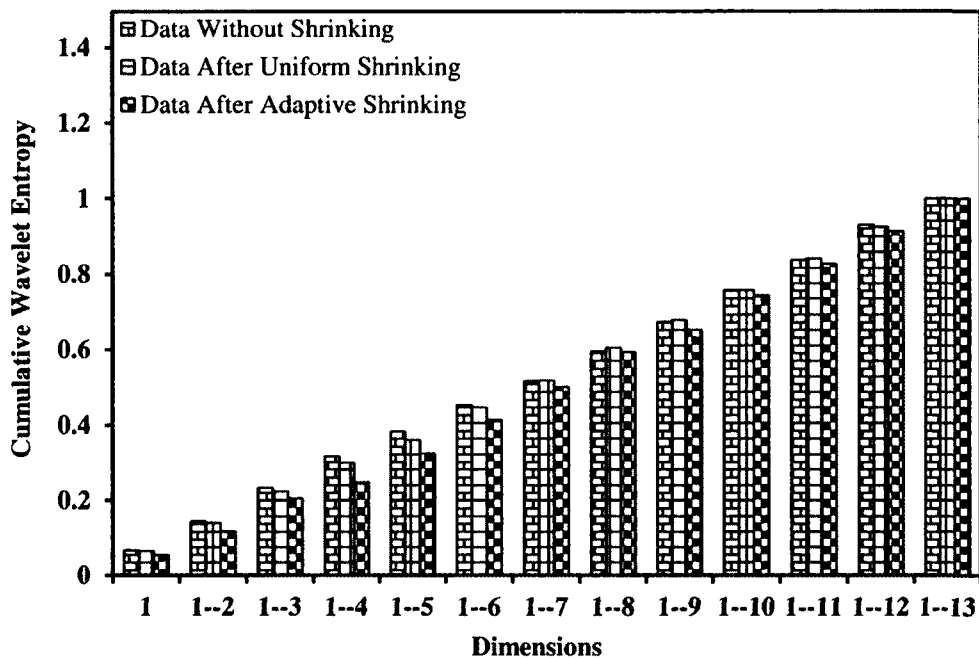


Figure 4.5: Cumulative Wavelet Entropy v/s Dimensions

Case 2: In this case, the energy of each dimension is computed in principal component space, and the percentage of the energy contributed by each dimension is

obtained. Finally, a plot is obtained that depicts the cumulative percentage of the energy for each set of dimension in principal component space. Figure 4.6 depicts a comparison of the cumulative energy. It can be observed from the plot that after non-uniform grid-based shrinking, dimensions retain the highest cumulative energy, which indicates that after performing non-uniform grid-based shrinking, each set of dimensions has more cumulative energy than the cumulative energy of each set of dimensions after performing uniform grid-based shrinking.

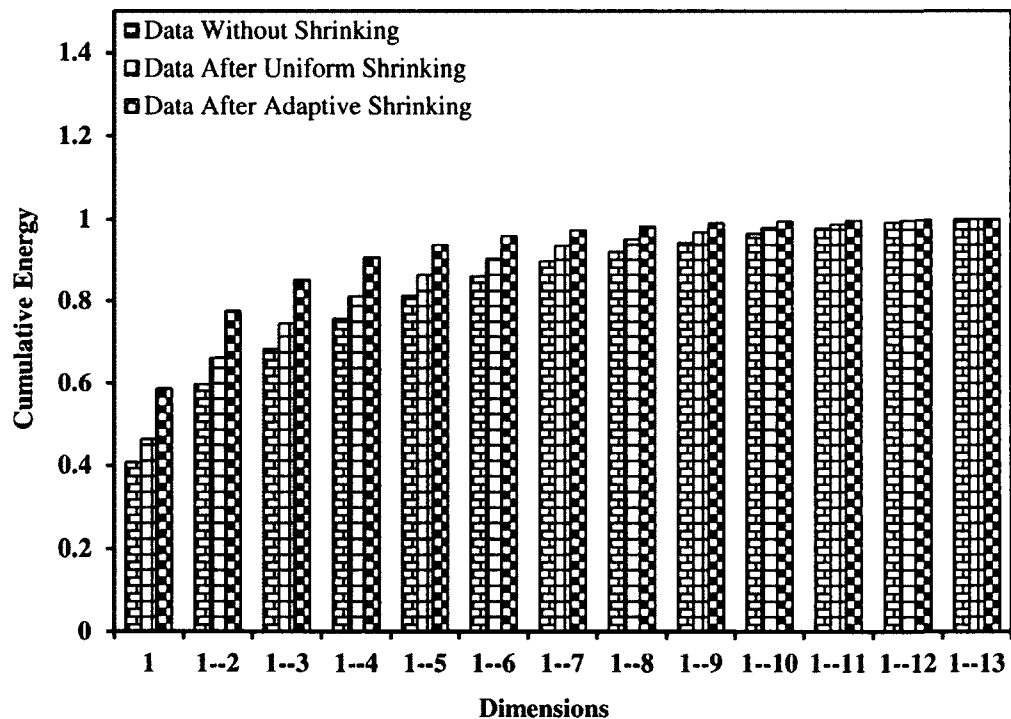


Figure 4.6: Cumulative Energy v/s Dimensions

Case 3: In this case, the information entropy of each dimension in principal component space is computed, and the percentage of the information entropy contributed by each dimension is obtained. Information entropy is a measure of disorder in the data

and its lower values are desired. Figure 4.7 depicts a comparison of the cumulative information entropy.

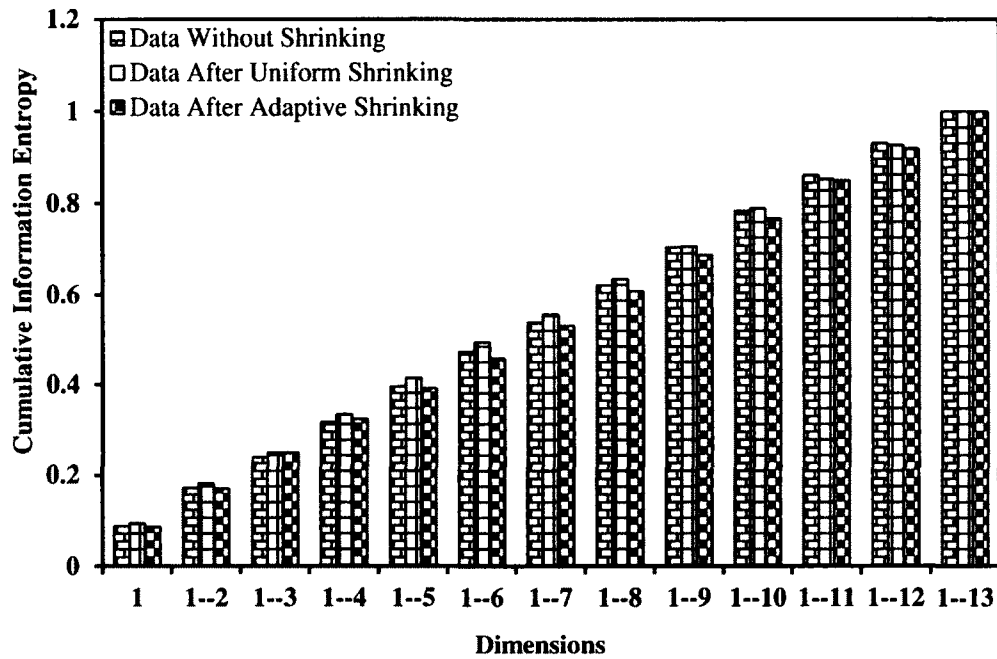


Figure 4.7: Cumulative Information Entropy v/s Dimensions

A comparative study is also conducted to demonstrate the computational benefits of non-uniform grid-based shrinking over the uniform grid-based shrinking. A plot of the comparative study is presented in Figure 4.8. In this comparative study, the average execution time of the uniform and the non-uniform grid-based shrinking is compared on a set of synthetic datasets. The uniform grid-based shrinking is performed using the algorithm presented in [13, 14]. Similarly, the non-uniform grid-based shrinking is performed using the algorithm presented in Chapter 6. To maintain constant experimental conditions, five iterations are performed on all the datasets for both shrinking algorithms.

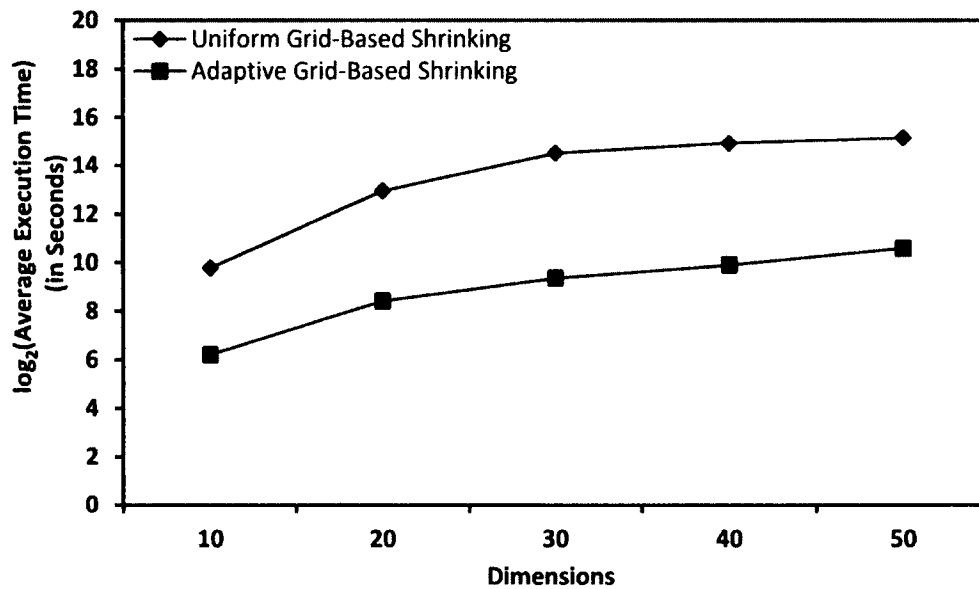


Figure 4.8: Average Execution Time v/s Dimensions

It can be inferred from the plot that non-uniform grid-based shrinking algorithm is computationally more efficient than uniform grid-based shrinking algorithm.

4.5 Conclusion

This chapter has highlighted the challenges of data preprocessing and emphasizes the need to develop better sparseness reduction algorithms to mitigate the detrimental effect of sparseness in multidimensional datasets. The limitations of existing sparseness reduction are also highlighted, and a need to develop a non-uniform grid-based shrinking approach is discussed. Furthermore, an experimental study is conducted to compare the uniform and the non-uniform grid-based partitioning and shrinking algorithms. The experimentations presented in Section 4.4 demonstrate that non-uniform grid-based partitioning and shrinking has a potential to be more effective than the existing uniform grid-based partitioning and shrinking algorithm.

CHAPTER 5

GRID-BASED LOCALIZED LEARNING FOR FEATURE RANKING

Many data preprocessing strategies have been proposed to sufficiently handle the high dimensionality of the data and avoid the infamous curse of dimensionality [2]. Dimensionality reduction methods, including feature selection, feature ranking, feature extraction, among other reduction strategies, have proven to be powerful in reducing this impediment [32, 33, 34, 35, 36, 37, 38, 39]. The underlying assumption of dimensionality reduction approaches is that not all dimensions are important, i.e. some dimensions may be irrelevant and detrimental to the efficacy of further data analysis, and hence can be eliminated. In feature selection and feature ranking, irrelevant features are eliminated from further consideration, thereby leaving only important features to be considered for further analysis. Furthermore, the feature ranking approaches use a scoring function to rank features according to their individual predictive power. Some common scoring functions are distance measures, information measures, dependency measures, and consistency measures [40, 41, 42, 43, 44, 45]. Most of the feature ranking methods rank each feature based on the feature's predictive power independently and ignore its dependency on other features. Thus, feature ranking methods are needed such that the feature ranking of an individual feature is also influenced by other features as well.

The remainder of the chapter is organized as follows. In Section 5.1 Research motivation is discussed. In Section 5.2, the problem statement and the hypothesis is discussed. In Section 5.3, developed feature ranking methodology is discussed. In Section 5.4, experimental study is discussed. Finally, in Section 5.5, the conclusions of this chapter are presented.

5.1 Research Motivation

Feature ranking approaches use a scoring function to rank features based on intrinsic data characteristics. Feature ranking approaches are preferable because of their low computational complexity and statistical scalability. Feature ranking methods use independent criteria or scoring functions to evaluate and rank individual features based on the predictive power of the feature and ignore any dependencies in the data. Thus, there is a lack of feature ranking or feature scoring functions that are influenced by the presence of other features in the data. In data shrinking, the movement of the data points changes the overall distribution of the data in multidimensional space as well as in individual dimensions. The difference in data distribution projected on every dimension through data shrinking can be captured by a shrinking profile of the dimension, and it can be used as a scoring function that is influenced by the presence of other dimensions in the data. Thus, the motivation for this research is to develop a new data shrinking based feature ranking algorithm to address the deficiencies of existing feature ranking techniques.

5.2 Problem Statement

Data shrinking is a data preprocessing technique that performs simulated movement of data points in multidimensional space, and data points move toward the center of their natural cluster [13, 14]. The movement of the data points changes the overall distribution of the data in the multidimensional space. The change in data distribution in a particular dimension is affected by the data distribution in every other dimension. Every dimension shrinks in a unique way, and some dimensions shrink more than others. Therefore, it is hypothesized that a scoring function based on the data shrinking can be used as a scoring function to measure the dimension's predictive power and can be utilized for feature ranking. Based on this hypothesis, the aim is to develop a framework that uses an adaptive grid-based data shrinking method for feature ranking.

5.3 Methodology

In this section, data shrinking based feature ranking framework is discussed. The developed feature ranking framework has four components. The first component is data preprocessing phase. The second component is data adaptive grid generation phase. The third component is the data adaptive grid-based shrinking phase and the fourth and final phase is the feature ranking and selection phase. All the components of data shrinking based feature ranking framework are explained below. The methodology is discussed as follows. In Section 5.3.1, data preprocessing operations applied on data are discussed. In Section 5.3.2, adaptive grid generation algorithm is discussed. In Section 5.3.3, the data shrinking algorithm is discussed. Finally, in Section 5.3.4 developed feature ranking framework is discussed. Figure 5.1 depicts the data shrinking based feature ranking framework that has been applied to a protein dataset.

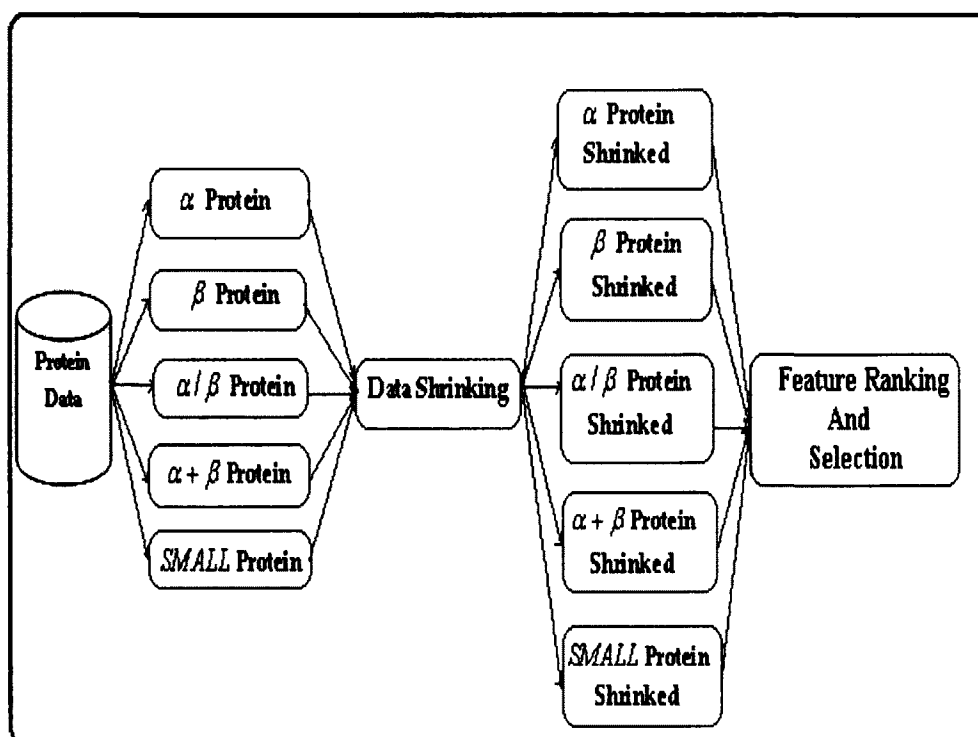


Figure 5.1: Data Shrinking Based Feature Ranking Framework

5.3.1 Data Preprocessing

Data preprocessing is an essential step in this methodology. The dataset is first standardized by applying Z-score normalization. Each dimension is transformed based on the mean and standard deviation of the dimension. The data is further normalized into a unit hypercube $[0, 1]^d$ to scale all the dimensions between the range of zero and one by applying min-max normalization on each dimension. In addition to this, those dimensions are eliminated from the datasets that do not provide significant variability within the dimension. It refers to the situation in which significant numbers of data values in a dimension are either zero or constant.

5.3.2 Adaptive Grid Generation

Grid structure is critical in grid-based data shrinking. With that in mind, a grid structure generation algorithm has been developed to utilize inherent data distribution

characteristics and generate adaptive grid boundaries for each dimension. The grid boundaries are determined by a wavelet transform based coefficient aggregation approach for the data adaptive grid structure. Initially, data is normalized in the unit hypercube $[0, 1]^d$, assuming there are d dimensions in the data. Figure 5.2 shows the algorithm that is applied for grid generation. The following procedure is followed for the generation of grid boundaries for single dimension and is then applied for all the other dimensions independently.

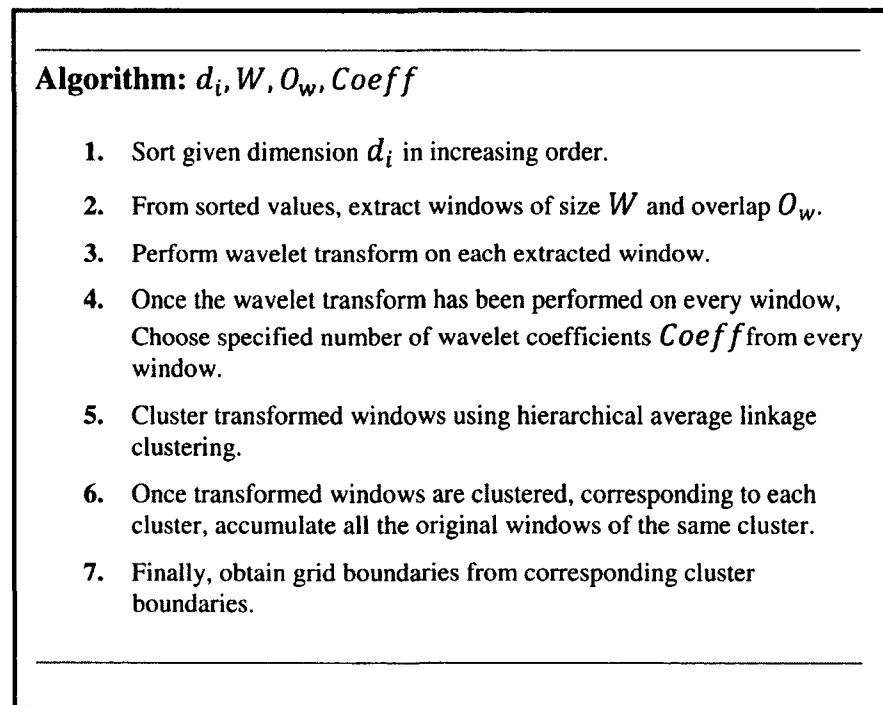


Figure 5.2: Adaptive Grid Generation Algorithm

5.3.3 Data Shrinking

The data adaptive grid-based shrinking algorithm begins once data adaptive partitions are obtained for all the dimensions. All the steps of the data shrinking algorithm are discussed below.

5.3.3.1 Data Movement Model

A grid-based model of attraction is employed for data movement. Let C_u be a grid cell that contains a set \mathbb{X}_u of k data points $\mathbb{X}_u = \{\vec{x}_{u1}, \dots, \vec{x}_{uk}\}$, where $\mathbb{X}_u \subset \mathbb{X}$ for which data movement is to be performed. Let $C_{NBR} = (C_{n1}, C_{n2}, \dots, C_{ni})$ be a set of neighboring grid cells that have (n_1, n_2, \dots, n_i) number of data points. Let the data centroid of all the data points in the set C_{NBR} of grid cells be given by the Equation 5.1. Similarly, the data centroid of all the points in the grid cell C_u is given by Equation 5.2:

$$\vec{c}_{NBR} = \frac{\sum_{j=1}^{n_i} \left(\frac{1}{n_j} (\sum_{r=1}^{n_j} x_r) \right)}{\sum_{j=1}^{n_j} n_j}, \quad \text{Eq. 5.1}$$

$$\vec{c}_u = \frac{1}{k} (\sum_{i=1}^k x_{ui}). \quad \text{Eq. 5.2}$$

Therefore, the movement or the displacement of a data point \vec{x}_{ui} in the grid cell C_u is given by Equation 5.3 below:

$$\vec{x}'_{ui} = \vec{x}_{ui} + (\vec{c}_{NBR} - \vec{c}_u). \quad \text{Eq. 5.3}$$

The movement or displacement of all the other data points in the grid cell C_u is performed. The movement of the data points is performed if it satisfies the movement threshold criteria given by Equation 5.4:

$$\text{Distance}(\vec{c}_{NBR}, \vec{c}_u) \geq M_{Th}. \quad \text{Eq.5.4}$$

The movement of data points is explained below. All the data points in a particular grid cell are moved as a single unit. First, identify all of grid cells C_u 's nonempty neighboring grid cells. Second, compute the data centroid of the selected neighboring grid cells of the grid cell C_u and the data centroid of the grid cell C_u . Third, move all the data points in the grid cell C_u using the data displacement formula presented

in Equation 5.1. This process is repeated for all the grid cells that have data points in them.

5.3.3.2 Data Shrinking Process

The algorithm first maps all data points on the adaptive grid. The pseudo-code of the algorithm is presented in Figure 5.3.

```

Algorithm: Data Shrinking Algorithm
Input: Grid  $G_1$ , Dataset  $\mathbb{X}$ , Iterations  $I_{Th}$ , Threshold  $M_{Th}$ 
Output: Data after Shrinking  $\mathbb{X}_1$ 
01  $N = \text{Number of Datapoints in } \mathbb{X}$ 
02  $d = \text{Number of Dimensions in } \mathbb{X}$ 
03 for  $i=1$  to  $N$ 
04    $C(i) = \text{Find\_Cell\_Id}(\vec{X}_i, G_1)$ 
05   Add  $C(i)$  to  $Z$ 
06   Add  $X_i$  to  $Zdata(\text{Count}).data$ 
07 end
08  $l = 0$ 
09 while  $l \leq I_{Th}$ 
10   for  $m=1$  to  $\text{length}(Z)$ 
11      $V(m) = \text{Compute\_Volume}(Z(m))$ 
12      $Rho(m) = \text{Compute\_Cell\_Density}(Z(m), V(m))$ 
13      $DenseZ = \text{Find\_Dense\_Cells}(Rho(m))$ 
14   end
15    $n = 1$ 
16   while  $n \leq \text{length}(DenseZ)$ 
17     Find Neighboring Cells of Cell  $DenseZ(n)$ 
18     Compute Centroid  $\vec{c}_{NBR}$  of Neighboring Cells
19     Compute Centroid  $\vec{c}_n$  of Cell  $DenseZ(n)$ 
20     if ( $\text{Distance}(\vec{c}_{NBR}, \vec{c}_n) \geq M_{Th}$ ) then
21       Compute Displacement of Datapoints in  $Zdata(n)$ 
22     end
23   end
24   if(No Movement between  $l$  and  $l + 1$ ) then
25     Exit
26   end
27    $Z = DenseZ$ 
28 end

```

Figure 5.3: Adaptive Data Shrinking Algorithm

During this process, it identifies all the non-empty grid cells and corresponding data points. It then accumulates all data points that are mapped to the non-empty grid cells. Next, volume and density of the nonempty grid cells which are populated with data points are computed. The density of a cell is defined as a fraction of the total number of data points in the cell over the cell volume. The volume of a grid cell is defined as a product of the side length of the grid cell over all the dimensions. Density threshold is used to identify dense cells and to discard others. Next, a dense cell is taken from the list of dense grid cells and its surrounding cells (that share an edge or a vertex with this cell) are captured in an adhoc cluster. The centroid of this cluster is computed. Then, all the data points in the grid cell are moved in the surrounding grid cells based on the model of data movement. This process is repeated for all the dense cells. The algorithm terminates after a specified number of iterations, or if termination criterion is satisfied.

5.3.4 Feature Ranking Method

The developed feature ranking algorithm is a two-step process. In the first step, feature weights are assigned based on their shrinking profile. In the second step, the features are ranked and selected based on their weights. Both the steps are as follows. The pseudo-code of the algorithm is presented in Figure 5.4.

The first step begins by computing a shrinking profile corresponding to each feature. The shrinking profile is computed by calculating the percentage change in mean square distance between all pairs of data points before shrinking and after shrinking. Next, weights are assigned to each feature using their shrinking profile and this process is repeated for each class separately. For this purpose, shrinking profiles of all the features are normalized and it is repeated for all the configurations of the algorithmic parameters.

Finally, a cumulative weight is obtained for each feature by summing all the weights across different configurations of algorithmic parameters.

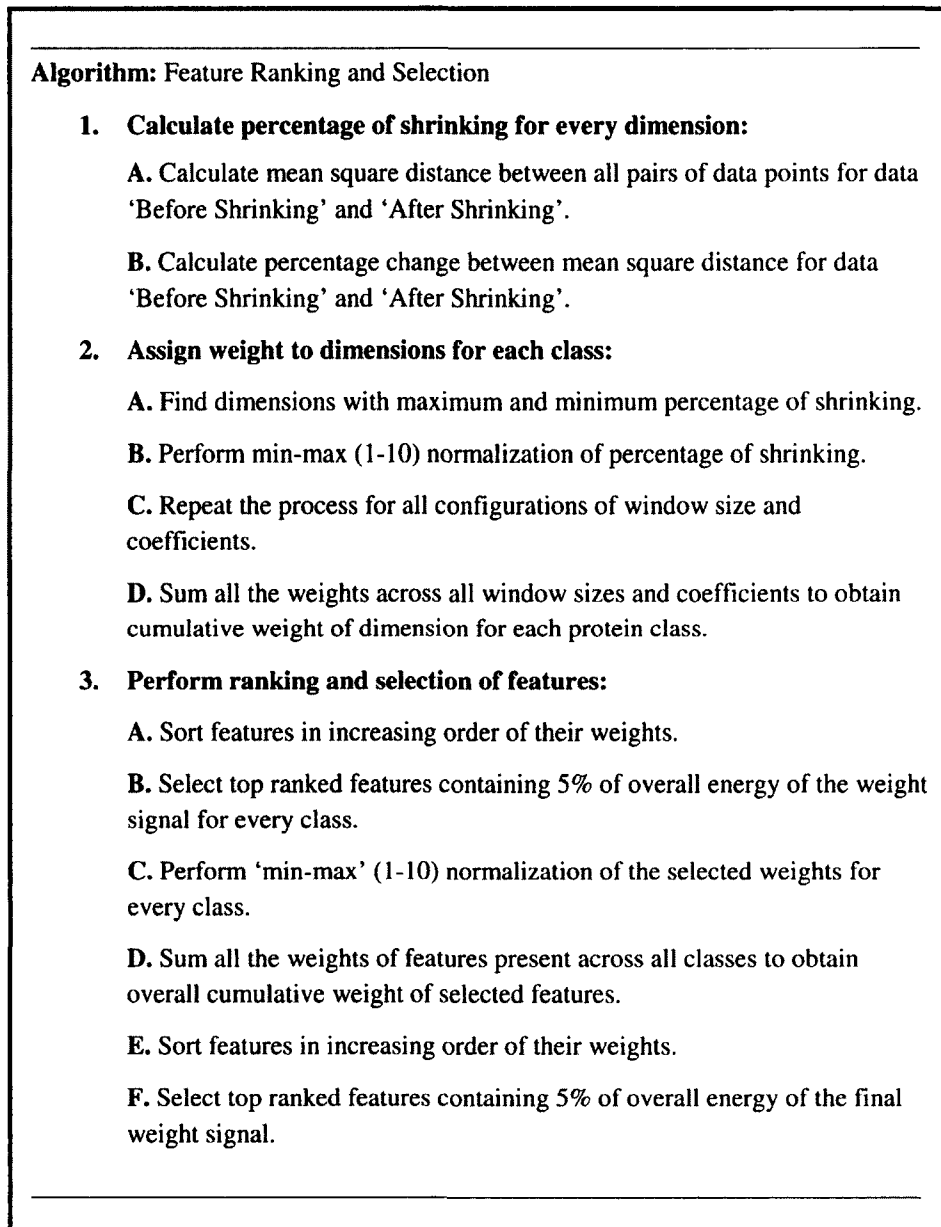


Figure 5.4: Feature Ranking and Selection Algorithm

The second step begins by sorting normalized feature weights in increasing order of their weights. Next, top ranked features are selected that contain only 5% of the total weight. Next, selected feature weights are normalized. This process is repeated for all the

features in their respective classes. Finally, a cumulative weight is obtained for all selected feature. These weights are sorted and final top ranked features are selected that contain only 5% of the total weight.

5.4 Results and Discussions

A set of experiments is performed to validate the developed data shrinking based feature ranking framework. Data shrinking is first performed on individual protein classes, and then feature ranking and selection is performed. A set of comparative study is conducted using different classifiers and different feature ranking methods to evaluate the feature ranking method. The remainder of the section is organized as follows. In Section 5.4.1, a brief description of all the datasets used for the experiments is given. In Section 5.4.2, validation technique and validation measures are discussed. Finally, in Section 5.4.3, experiments related to the comparative analysis are presented.

5.4.1 Datasets

Experiments are conducted on a high dimensional proteomics dataset. Proteomics is high throughput data discipline, and multidimensionality is an inherent characteristic of the proteomic data. For example, hundreds of feature descriptors may be generated from the physiochemical properties of the proteins [59, 60]. Proteomics dataset is characteristically high dimensional and exhibits sparseness. Therefore, a protein dataset is chosen for experiments that have been used in the past. The protein dataset contains both a training dataset and a test dataset and consists of five protein structural classes and 125 feature descriptors. The training data has 408 training samples, and the test dataset has 174 test samples from five protein structural classes. The features of the dataset are

extracted from the protein sequence information using the method discussed in [59, 60]. This data is available at this URL (<http://ranger.uta.edu/~chqding>).

5.4.2 Validation

Validation of the developed feature ranking method is done by comparing it with other existing feature ranking methods. For the purpose of validation, the RELIEF algorithm, the Chi-Square filter, the information gain based method, and SVM based feature ranking method are used [43, 61, 62, 63, 64, 65]. The developed feature ranking algorithm is compared with other well-known feature ranking algorithms based on their performance on classification methods. The classification performance of these methods is assessed through external validation measures precision, recall, F-measure, and classification accuracy. These measures are represented by Equations 5.5, 5.6, 5.7, and 5.8:

$$Precision = \frac{TP}{(TP+FP)}, \quad \text{Eq. 5.5}$$

$$Recall = \frac{TP}{(TP+FN)}, \quad \text{Eq. 5.6}$$

$$F - measure = 2 \times \left(\frac{precision+recall}{precision+recall} \right), \quad \text{Eq. 5.7}$$

$$Classification Accuracy = \frac{(TP + TN)}{(TP+FP+TN+FN)}, \quad \text{Eq. 5.8}$$

In the above equations, TP, TN, FP, FN refer to true positive, true negative, false positive, and false negative, respectively.

5.4.3 Experiments

Experiments are conducted on the datasets to demonstrate that the proposed method is capable of effective feature ranking and selection. To demonstrate that the feature ranking method works effectively, it is compared with classical feature ranking

methods. Figure 5.5 also displays a comparative analysis of the common Top-10, Top-20, Top-30, and Top-41 features with other existing feature ranking methods.

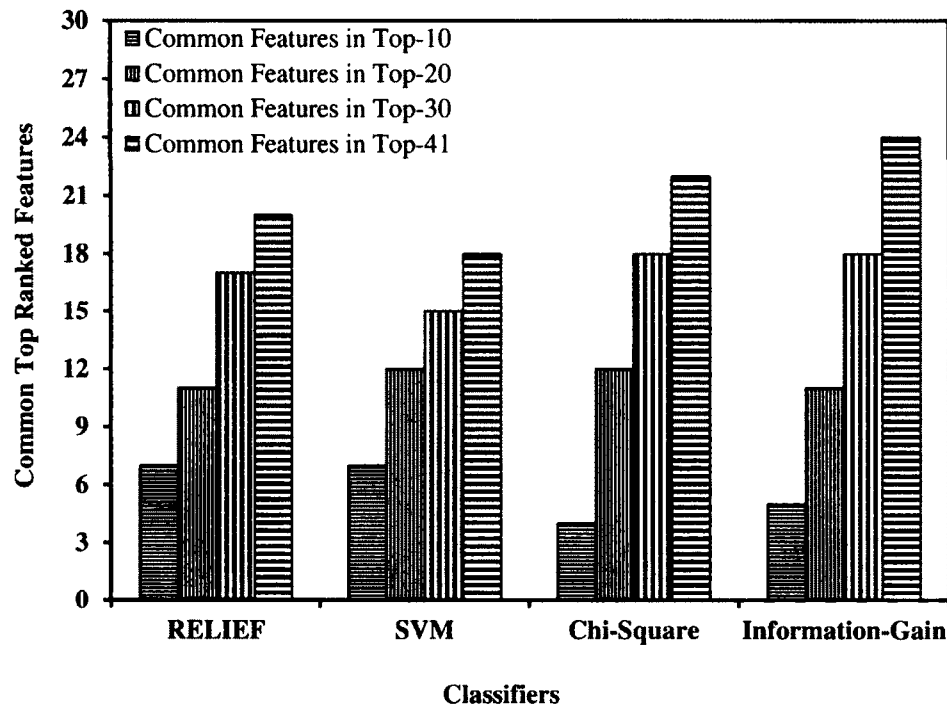


Figure 5.5: Comparison of Top Ranked Features

Table 5.1 displays the common top ranked features. Table 5.1 displays the top ranked 41 features for the comparative methods and for the data shrinking based feature ranking method. Table 5.1 shows approximately 45%-60% of feature commonality between the top 41 ranked features (those indicated in bold) and the top ranked features in comparative methods.

Table 5.1: Common Top Ranked Features

Feature Selection	Top ranked 41 features
Information Gain	84,85,88,95,89,100,86,92,94,2,96,93,90,98,97,91,99,105,110,18,27,121,79,111,32,69,116,1,106,53,6,37,48,58,74,19,52,87,36,10,109
Chi Square	84,2,86,100,88,85,89,95,96,92,94,93,90,18,97,98,110,91,79,121,105,106,99,27,6,116,120,118,37,111,1,36,32,69,109,48,53,117,10,19,119
SVM	84,94,121,2,95,118,109,99,85,90,43,86,27,1,89,105,6,106,18,17,125,79,110,9,64,16,78,8,96,13,36,70,33,62,37,93,32,42,15,19,40
RELIEF	84,94,99,93,88,98,89,92,85,2,86,97,95,91,96,90,1,18,106,100,105,110,32,27,87,9,58,37,5,109,6,48,28,11,121,79,64,8,59,10,13
Shrinking Method	104,62,58,31,94,37,27,48,99,74,91,90,32,85,17,102,10,83,125,69,103,111,78,73,89,115,49,87,98,18,120,80,122,2,93,95,116,57,79,121,79

The performance of data shrinking based feature ranking framework is compared with other existing feature ranking methods. The strength of all the feature ranking method is evaluated against a set of classifiers. Classification results of data shrinking based feature ranking framework is compared with information gain based feature ranking, and χ^2 feature ranking method [43, 64]. The classifiers that are used for comparison include, PART rule based classifier, Logistic regression and Neural Network [66, 67, 2]. This comparative analysis is conducted on protein data that has separate training and test set.

1. Comparative Study of F-measure:

The F-measure of shrinking based feature ranking algorithm are compared with the χ^2 method and information gain based feature ranking on the neural network classifier. In Table 5.2, the values of F-measure are compared over all the protein classes.

Table 5.2: Comparison of F-measure for Neural Network

Classifier	Feature Selection Method	Chi-Square Method	Info. Gain Method	Shrinking Method
Neural Network	Classes	F-measure (%)	F-measure (%)	F-measure (%)
	Class α	78.90	82.90	84.80
	Class β	58.20	69.60	74.10
	Class α/β	74.80	76.10	76.10
	Class $\alpha + \beta$	35.30	30.80	44.40
	Class <i>Small</i>	100.00	100.00	100.00

After comparing the values of the F-measure for each protein class, it can be said that shrinking based feature ranking either outperforms or gives comparable results. If the average F-measure value over all the class is compared, then the average F-measure values for χ^2 method and information gain method and shrinking based method are 69.44%, 71.88%, and 75.88%, respectively. This comparison indicates that shrinking based feature ranking performs better than the other two methods.

Similarly, the F-measure values of shrinking based feature ranking, χ^2 method and information gain based feature ranking are compared for a rule based classifier PART. In Table 5.3, the values of F-measure are compared over all the classes and it can be said that the shrinking based feature ranking gives comparable results. If the average F-measure value over all the class is compared, then the average F-measure values for χ^2 method and information gain method and shrinking based method are 68.64% 73.64%, and 75.94%, respectively. This comparison indicates that shrinking based feature ranking performs better than the other two methods for rule based classifier PART [66].

Table 5.3: Comparison of F-measure for PART

Classifier	Feature Selection Method	Chi-Square Method	Info. Gain Method	Shrinking Method
PART	Classes	F-measure (%)	F-measure (%)	F-measure (%)
	Class α	75.30	79.40	84.40
	Class β	72.70	72.00	67.80
	Class α/β	74.10	73.30	73.00
	Class $\alpha + \beta$	21.10	43.50	54.50
	Class <i>Small</i>	100.00	100.00	100.00

The F-measure values of shrinking based feature ranking, χ^2 method and information gain based feature ranking are also compared for logistic regression based classifier. In Table 5.4, the values of F-measure are compared over all the classes. It is observed from the table that shrinking based feature ranking outperforms or gives comparable results. The average F-measure value over all the class is also compared. The average F-measure values for χ^2 method and information gain method and shrinking based method are 57.50%, 53.06%, and 64.66%, respectively. This comparison indicates that shrinking based feature ranking performs better than the other two methods for logistic regression classifier [67].

Table 5.4: Comparison of F-measure for Logistic Regression

Classifier	Feature Selection Method	Chi-Square Method	Info. Gain Method	Shrinking Method
Logistic Regression	Classes	F-measure (%)	F-measure (%)	F-measure (%)
	Class α	62.90	52.70	70.00
	Class β	56.80	44.40	66.10
	Class α/β	71.80	66.10	69.80
	Class $\alpha + \beta$	0.00	6.10	21.40
	Class <i>Small</i>	96.00	96.00	96.00

2. Comparative Study of Average Precision, Recall, and Accuracy:

A comparison of average precision, recall and accuracy are also conducted to compare shrinking based feature ranking algorithm, χ^2 method and information gain based ranking method on the rule based classifier PART. In Table 5.5, a comparison of the average values of precision, recall and classification accuracy is presented.

Table 5.5: Comparison of Avg. Precision, Recall, Accuracy for PART

Classifier	PART		
	Average Recall (%)	Average Precision (%)	Overall Accuracy (%)
All Features	71.60	70.54	72.41
Chi-Square Method	69.80	70.86	72.99
Info. Gain Method	73.00	75.58	74.14
Shrinking Method	74.20	80.04	74.14

In Table 5.5, after comparing the average values of precision, recall and accuracy, it can be said that shrinking based feature ranking either outperforms or gives comparable results for all the measures.

Similarly, a comparison of average precision, recall and accuracy is also performed to compare shrinking based feature ranking algorithm, χ^2 method and information gain based ranking method on the logistic regression classifier [67]. It is presented in Table 5.6. In Table 5.6, after comparing the average values of precision, recall and accuracy, it can be concluded that shrinking based feature ranking method gives superior and comparable results when compared with other methods.

Table 5.6: Comparison of Avg. Precision, Recall, Accuracy for Logistic Regression

Classifier	Logistic Regression		
	Average Recall (%)	Average Precision (%)	Overall Accuracy (%)
All Features	63.60	74.38	67.24
Chi-Square Method	60.20	57.70	64.37
Info. Gain Method	53.80	55.16	53.45
Shrinking Method	65.40	65.50	66.67

Additionally, a comparison of average precision, recall and accuracy is also performed to compare shrinking based feature ranking algorithm, χ^2 method and information gain based ranking method on the neural network classifier. It is presented in Table 5.7. In Table 5.7, after comparing the average values of precision, recall and accuracy, it can be concluded that shrinking based feature ranking outperforms other

methods on average accuracy and recall. However, when compared with other methods on average precision, it does not give good comparable results.

Table 5.7: Comparison of Avg. Precision, Recall, Accuracy for Neural Network

Classifier	Neural Network		
Feature Selection Method	Average Recall (%)	Average Precision (%)	Overall Accuracy (%)
All Features	72.40	80.60	72.98
Chi-Square Method	68.80	84.80	71.84
Info. Gain Method	72.20	72.40	74.14
Shrinking Method	76.00	76.40	76.44

5.5 Conclusion

In this work a data shrinking based novel approach of feature ranking and selection have been presented. Every dimension participates in the shrinking process, but every dimension shrinks differently. Some shrink a great deal; others shrink only a little. Thus, the way the dimension shrinks decides its characteristics. These characteristics are used to find the most discriminating features. The experimental study suggests that features that shrink less exhibit good discriminating behavior. The results confirm this hypothesis.

CHAPTER 6

GRID-BASED LOCALIZED LEARNING FOR CLASSIFICATION

The increase in the demand for data mining algorithms that are fast, scalable and accurate has resulted in the development of scalable classification models [48, 49, 50, 51]. Scalability, a central component in the design of a scalable classifier, refers to an algorithm's ability to handle the increase in the size and dimensionality of the dataset. A scalable classifier should scale well; i.e. its performance should not deteriorate drastically with the increased dataset size and dimensionality of the dataset. However, the existing classification algorithms that perform well for the small and medium dimension datasets fail to perform well when the dimensionality and size of the datasets increase. Therefore, there is a need to develop new classification methods that are fast, scalable and accurate.

The remainder of the chapter is organized as follows. In Section 6.1, research motivation is discussed. In Section 6.2, the problem statement is discussed. In Section 6.3, the methodology of the grid-based classification models is discussed. In Section 6.4, experimental results are discussed. Finally, in Section 6.5 conclusions and future directions are discussed.

6.1 Research Motivation

The potential of grid-based localized learning is well recognized in unsupervised learning algorithms [2]. However, the potential of grid-based localized learning has not been exploited adequately in designing supervised learning algorithms. The grid-based localized learning algorithms can scale well with an increase in the dimensionality and size of datasets. The grid-based localized learning algorithms are inherently scalable because they reduce the search space by partitioning the feature space into uniform or non-uniform partitions [6, 7, 8, 9, 10, 11, 12]. Thus, the motivation is to develop grid-based classification models to harness the scalable nature of the grid-based localized learning paradigm.

6.2 Problem Statement

Grid-based localized learning paradigm has been used in designing fast and scalable unsupervised learning algorithms that scale well with respect to the increase in size of the dataset and dimensionality of the dataset. Therefore, it is hypothesized that grid-based classification models can be developed using the grid-based nearest-neighbor learning approach to develop fast and scalable classification models. Based on this hypothesis, the aim is to develop the grid-based classification models that inherit the advantages of grid-based localized learning paradigm.

6.3 Methodology

In this section, fixed grid-based and adaptive grid-based classification models are discussed. The developed grid-based classification models consist of four phases. The first phase is the data preprocessing phase. The second phase is the grid generation phase.

The third phase is the training phase of the classifier design and the fourth phase is the test phase of the classifier design. All the phases of fixed grid-based classifier and adaptive grid-based classifier are identical except the grid generation phase. All the phases of the methodology are as follows.

6.3.1 Data Preprocessing

Data preprocessing is an essential step in this methodology. The dataset is first normalized by applying Z-score normalization. Each dimension is transformed based on the mean and standard deviation of the dimension. The data is further normalized into a unit hypercube $[0, 1]^d$ to scale all the dimensions between the range of zero and one by applying min-max normalization on each dimension.

6.3.2 Grid Generation

Grid generation is essential for the grid-based classification model. Two methods of grid generation are discussed here. The first method generates uniform grid structure and the second method generates adaptive grid structure.

6.3.2.1 Uniform Grid Generation

Uniform grid structures are generated by creating uniform partitions of the desired size in each dimension. The uniform grid generation is a simple process. First, it is assumed that data is normalized between $[0, 1]$. Next, each dimension is partitioned into equal width of the desired number of partitions. The partition width is given by

Equation 6.1:

$$Mp_{Size} = \frac{1}{Mp_{Number}}. \quad \text{Eq. 6.1}$$

Here, Mp_{Number} represents user defined number of partitions and Mp_{Size} represents the size of each partition.

6.3.2.2 Adaptive Grid Generation

The adaptive grid structures are essential for the adaptive grid-based classification model. Therefore, an algorithm is developed that generates adaptive grids by creating data adaptive partitions in each dimension. The adaptive grid generation is a two-step process. First, each dimension is sorted and micro-partitions (see Definition 3.18) are created. Next, micro-partitions are clustered using the minimum variance based selective agglomerative hierarchical partitioning. The following steps are performed on each dimension to generate adaptive grid.

Creating Micro-Partitions: Initially, the dimension is sorted in ascending order. The sorted one-dimensional data points are in close proximity with their neighbors. Non-overlapping units of data points called micro-partitions are created by grouping k contiguous data points ($k < N$, where N is the total number of data points). A small value of k is chosen because micro-partitions should be as small as possible but not small enough to undermine the benefits of the overall grid generation process. The choice for the size of micro-partitions is inspired by [68, 69]. The size of a micro-partition is obtained by applying Equation 6.2 and the number of micro-partitions is obtained by applying Equation 6.3:

$$Mp_{Size} = \left\lceil \sqrt{N/10} \right\rceil, \quad \text{Eq. 6.2}$$

$$Mp_{Number} = \left\lfloor \frac{N}{Mp_{Size}} \right\rfloor. \quad \text{Eq. 6.3}$$

Choosing a value smaller than $N/10$ will reduce the size of micro-partitions and will create too small micro-partitions and undermine the benefits of micro-partitioning and will have higher computational cost.

Variance-Based Partitioning: The variance-based hierarchical partitioning groups contiguous micro-partitions in bottom-up fashion. See Figure 6.1 for the pseudo-code of the algorithm.

Algorithm: Variance Based Partitioning
Input: Dataset \mathbb{X}
Output: Data Adaptive Grid G

```

01  $N = \text{Number of Datapoints in } \mathbb{X}$ 
02  $d = \text{Number of Dimensions in } \mathbb{X}$ 
03 for  $j=1$  to  $d$ 
04    $S_d = \text{Sort}(\mathcal{D}_j)$ 
05    $Mp_{Size} = \lceil \sqrt{N/10} \rceil$ 
06    $Mp_{Number} = \lfloor \frac{N}{Mp_{Size}} \rfloor$ 
07    $n = Mp_{Number}$ 
08 for  $r=1$  to  $(n - 1)$ 
09    $VAR(r) = \frac{1}{(n_r+n_{r+1}-1)} \sum_{i=1}^{n_r+n_{r+1}} |x(r, i) - \overline{x_{r,r+1}}|^2$ 
10 end
11 while  $n \neq 2$ 
12    $MergeIndex = \text{Find\_minimum}(VAR)$ 
13    $M_n = \text{Merge\_micro\_partitions}(MergeIndex)$ 
14    $n = n - 1$ 
15 end
16 for  $n=2$  to  $Mp_{Number}$ 
17    $M_n(m) = \text{Find\_partitions}(M_n)$ 
18 end
19 end

```

Figure 6.1: Variance-Based Partitioning Algorithm

The algorithm begins by creating micro-partitions of desired size, which is obtained using Equation 6.2. Next, the computation of the proximity between all the pairs of adjacent micro-partition is performed using the combined variance of adjacent micro-partitions. Two contiguous micro-partitions are grouped together based on the minimum combined variance. This process of grouping adjacent micro-partitions continue in bottom-up fashion until all the micro-partitions are grouped together in one big partition.

Once the algorithm is terminated, corresponding micro-partitions are grouped and a hierarchical decomposition of partitions is obtained for the dimension. This process is repeated for all the dimensions in the similar fashion.

6.3.3 Training Phase

The training phase of the classifier is a two-step process. In the first step, the training data is mapped on the fixed grid structure or the adaptive grid structure depending on the classification model. Every training sample is mapped on the grid structure by assigning every training sample to its corresponding grid cell. This step is called class mapping. In the second step, the grid-based neighborhood model is built by identifying the neighborhood of every nonempty grid cell. This step is called neighborhood identification. Both the steps are intertwined in the training phase and are discussed here.

Class Mapping: Class mapping is the process of assigning the training sample of a particular class to its corresponding grid cell. In this process, a given n -dimensional training sample is assigned to a corresponding cell by assigning each data value in a dimension to an appropriate partition of the dimension, thus identifying its cell ID. This cell ID is stored along with the training sample and its class label. For the next training sample the same process is applied and its grid cell is identified. The cell ID of the training sample is matched against previously added cell ID's. If a match is found, then this training sample and its class label is appended to the existing list. If no match is found, then this cell ID is added to the existing list of grid cells along with the training sample, its class label and the neighborhood information. This process is repeated for all the training samples. The pseudo-code of the training phase is presented in Figure 6.2.

Algorithm: Training Phase
Input: Training Dataset \mathbb{X} , Grid G_1
Output: Grid-Based Classification Model ($Z, Zdata, Zneighbor$)

```

01  $N = \text{Number of Datapoints in } \mathbb{X}$ 
02  $Count = 0$ 
03 for  $i=1$  to  $N$ 
04    $C(i) = \text{Find\_Cell\_Id}(\vec{X}_i, G_1)$ 
05    $NbrCount = 0$ 
06    $NeighborList = \emptyset$ 
07   for  $m=1$  to  $Count$ 
08     if ( $Z(m) == C(i)$ ) then
09        $Z(m).count = Z(m).count + 1$ 
10        $\text{Add } X_i \text{ to } Zdata(m)$ 
11       break
12     else
13       if ( $\text{Neighbor}(Z(m), C(i))$ ) then
14          $NbrCount = NbrCount + 1$ 
15          $\text{Add } m \text{ to } NeighborList$ 
16       end
17     end
18   end
19   if ( $C(i) \notin Z$ ) then
20      $Count = Count + 1$ 
21      $\text{Add } C(i) \text{ to } Z$ 
22      $\text{Add } X_i \text{ to } Zdata(Count)$ 
23      $\text{Add } NeighborList \text{ to } Zneighbor(Count)$ 
24     for  $n=1$  to  $NbrCount$ 
25        $\text{Add } Count \text{ to } Zneighbor(NeighborList(n))$ 
26     end
27   end
28 end

```

Figure 6.2: Training Phase of the Grid-Based Classifier

Neighborhood Identification: Neighborhood identification refers to the process of identifying the neighboring grid cells (see Definition 3.9) of a grid cell. The neighborhood of a grid cell is identified by matching the cell ID of the training sample against previously added cell ID's in the list of grid cells. If the match satisfies the neighborhood criterion, then the index of the previously added cell ID is added to the list of the neighboring grid cell otherwise it is not added to the list. This process is repeated for the entire list of grid cells. The neighborhood list of all the existing grid cells is also

updated after adding the newly identified grid cell to the list of grid cells. This process is repeated for every new grid cell that is added to the list of grid cells.

6.3.4 Test Phase

In the test phase, each n -dimensional test sample is assigned to its corresponding grid cell by assigning each data value in a dimension to an appropriate partition of the dimension, thus identifying its cell ID. This procedure is repeated for all the test samples. Initially, the grid cell ID of the test sample is matched against the list of grid cell ID's of training data. If a match is found, then the training samples and the neighboring grid cells corresponding to the matched grid cell are obtained. Next, the distance between the test sample and the medoid of the training samples present in each neighboring cell is computed and k -nearest-neighbors are identified. The test sample is assigned to the class that has the majority votes in the k -nearest-neighbors list. Furthermore, if no match is found, then the distance between the test sample and the medoid of the training samples present in a grid cell which is the element of the list of grid cell ID's of training data is computed. This process is repeated for each grid cell present in the list of grid cells ID's of training data. Finally, k -nearest-neighbors are identified and the test sample is assigned to the class that has the majority votes in the k -nearest-neighbors list. The pseudo-code of the test phase is presented in Figure 6.3.

```

Algorithm: Test Phase
Input: Test Dataset  $\mathbb{X}$ , Grid  $G_1$ ,  $Z$ ,  $Zdata$ ,  $Zneighbor$ 
Output: Predicted Class Labels  $PLabel$ 
01  $N = \text{Number of Datapoints in } \mathbb{X}$ 
02  $Count = \text{Number of Classes}$ 
03 for  $i = 1$  to  $N$ 
04    $C(i) = \text{Find\_Cell\_Id}(\vec{X}_i, G_1)$ 
05    $NbrCount = 0$ 
06    $NbrList = \emptyset$ 
07    $NbrData = \emptyset$ 
08   for  $m = 1$  to  $\text{length}(Z)$ 
09     if ( $Z(m) == C(i)$ ) then
10        $Cellindex = m$ 
11        $NbrList = Zneighbor(m)$ 
12        $NbrData = Zdata(m)$ 
13       break
14     else
15       if ( $Neighbor(Z(m), C(i))$ ) then
16          $NbrCount = NbrCount + 1$ 
17         Add  $m$  to  $NbrList$ 
18       end
19     end
20   end
21   for  $n = 1$  to  $\text{length}(NbrList)$ 
22      $KNNList = \text{Find\_KNearest Neighbor}$ 
23   end
24    $PLabel(i) = \text{Assign\_Class\_Label}$ 
25 end

```

Figure 6.3: Test Phase of the Grid-Based Classifier

6.4 Results and Discussions

The performance of the developed classifier is measured based on the time complexity of the classifier, scalability of the classifier and the correctness of the classifier. The time complexity of the classifier computes the time required by the classifier to build and test the model. The scalability of the classifier measures its time requirement with respect to the increasing dimensions and dataset size and the correctness of the classifier measures its ability to correctly classify the data. The remainder of the section is organized as follows. In Section 6.4.1, a brief description of

all the datasets used for the experiments is given. In Section 6.4.2, validation technique and validation measures are discussed. In Section 6.4.3, experiments pertaining to the scalability analysis and comparative analysis are presented. Finally, in Section 6.4.4 time complexity analysis of the classifier is presented.

6.4.1 Datasets

Both real and synthetic datasets with a wide range of dimensions and sample size are used for experiments and assess the capabilities of the classifier. A detailed description of each of these datasets is as follows:

- 1. Letter Recognition Dataset:** The letter recognition dataset consists of 16 primitive numerical features extracted from character images of 26 capital letters in the English alphabet. These numerical features are statistical moments and edge counts. The dataset has 20,000 sample images and 16 dimensions. The dataset has 26 classes and each class represents the 26 capital letters in the English alphabet. Each class has approximately 700 to 800 data samples. The dataset is available at the UCI data archive website (<http://archive.ics.uci.edu/ml/datasets.html>) [58].
- 2. Handwritten Numeral Recognition Dataset:** The handwritten numerals recognition dataset consists of features extracted from the binary images of the ten numerals (0-9) that were obtained from a collection of Dutch utility maps. There are 200 samples per numeral and a total of 2000 samples overall. The dataset has 10 classes and each class represents 10 numerals. A feature set extracted from the binary images is used for experiments. The feature set consists of profile correlations of binary images and it has 216 dimensions. The dataset is available at the UCI data archive website (<http://archive.ics.uci.edu/ml/datasets.html>) [58].

- 3. Protein Structural Classification Dataset:** The protein dataset consists of feature vectors that are based on amino acid sequence of corresponding proteins. The feature construction is based on the amino acid composition, physical, and stereo chemical properties of amino acids. Each feature vector consists of 125 feature descriptor. The dataset has 582 samples and is divided into five protein structural classes, namely α , β , $\alpha + \beta$, $\frac{\alpha}{\beta}$, and *Small* proteins. The feature vector construction method is discussed in [60]. This data is available at (<http://ranger.uta.edu/~chqding/protein/>).
- 4. Synthetic Dataset:** Synthetic datasets are generated for the experiments pertaining to the scalability study. A set of 20 synthetic datasets is generated that consist of all the combinations of 25, 50, 75, 100, and 125 dimensions, and 10000, 20000, 30000, and 40000 data points. Each dataset contains four clusters, each of which has an equal number of data points in respective datasets. A R package is used for generating the synthetic datasets with the desired degree of separation [70]. The value of separation index ranges between -1 to 1. A value of separation index closer to one indicates that all the clusters are well separated.

6.4.2 Validation

The ability of the classifier to correctly identify the test samples is performed either by holdout method or a variant of k-fold cross-validation. Stratified k-fold cross-validation which is a variant of standard k-fold cross-validation is used to estimate the classification performance of the classifier on all the datasets. Each class is divided into k disjoint subsets and approximately equal in size. In k-fold cross-validation, k-1 folds are used for training the classifier and the remaining one fold is used for evaluating the classifier. This process is repeated k times, leaving one different fold for evaluation each

time. Classification accuracy and F-measure is used to compare the classification results. F-measure is the combination of precision and recall measure. These measures are represented by Equations 6.4, 6.5, 6.6, and 6.7:

$$Precision = \frac{TP}{(TP+FP)}, \quad \text{Eq. 6.4}$$

$$Recall = \frac{TP}{(TP+FN)}, \quad \text{Eq. 6.5}$$

$$F - measure = 2 \times \left(\frac{precision \cdot recall}{precision + recall} \right), \quad \text{Eq. 6.6}$$

$$Classification Accuracy = \frac{(TP + TN)}{(TP+FP+TN+FN)}. \quad \text{Eq. 6.7}$$

In the above equations, TP, TN, FP, and FN refer to true positive, true negative, false positive, and false negative, respectively.

6.4.3 Experiments

This section initially discusses the validation technique and validation measures used to evaluate the classifier. Then experiments are presented to demonstrate the scalability of the classifier with increasing dimensions and dataset size. Finally, experiments are presented to compare the ability of the developed grid-based classifier with other existing classifier to correctly identify the test samples.

6.4.3.1 Scalability Analysis

Experiments are conducted to establish the scalability characteristics of the developed classifiers. The scalability study of the training phase and the test phase of the fixed grid-based and adaptive grid-based classifier are presented below. The time requirements of the training phase and the test phase of both the classifiers do not deteriorate drastically and an appearance of linearity is observed as the number of dimensions and the size of the datasets increase.

Fixed Grid-Based Classifier: The scalability study pertaining to the training phase and the test phase of the classifier is as follows:

Training Phase: Figure 6.4 plots the execution time of the training phase of the classifier on a set of twenty synthetic datasets. This figure demonstrates the scalability of the training phase of the classifier with respect to the increasing dimensions of the dataset while keeping the data size constant. It is demonstrated from Figure 6.4 that the training time required by the classifier appears to increase linearly with the increase in dimensions. Similarly, Figure 6.5 depicts the scalability of the training phase of the classifier with respect to the increasing size of the dataset while keeping the dimensions constant. Figure 6.5 plots the execution time of the training phase of the classifier with respect to the size of the dataset. It is demonstrated from the figure that the time required by the classifier appears to increase linearly with the increase in the size of the dataset.

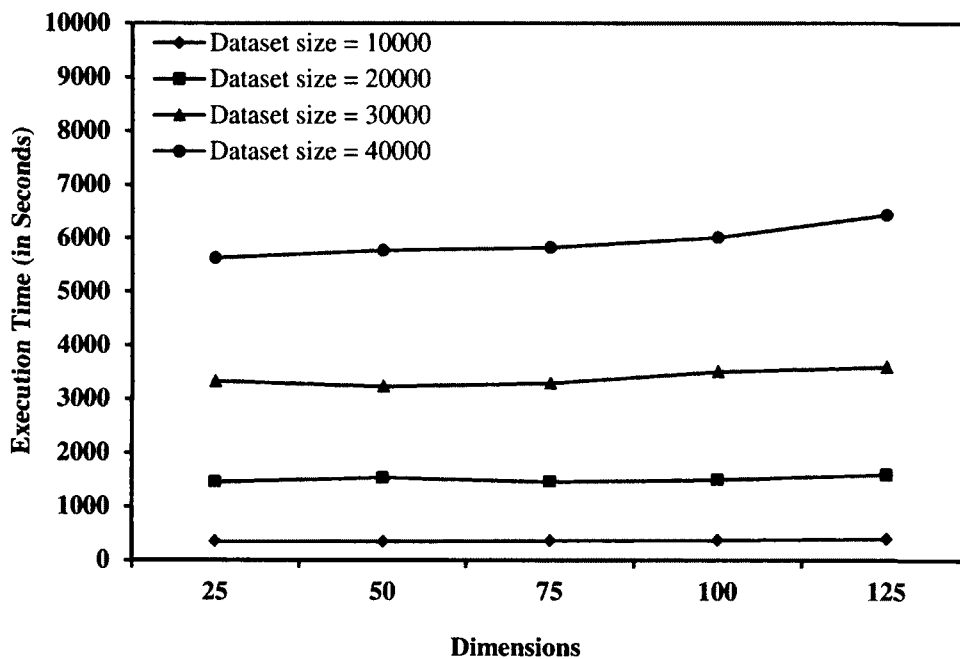


Figure 6.4: Training Phase Execution Time v/s Dimensions

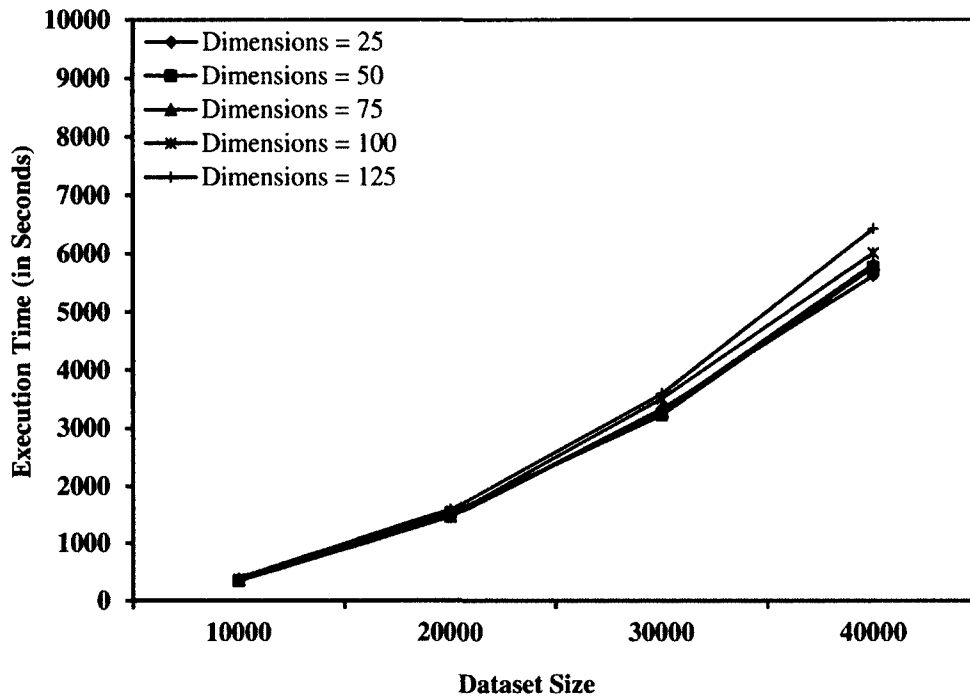


Figure 6.5: Training Phase Execution Time v/s Dataset Size

Test Phase: Figure 6.6 demonstrates the scalability of the test phase of the classifier with respect to the increasing dimensions of the dataset while keeping the data size constant. Figure 6.6 depicts the average execution time taken by each test sample with respect to the increasing dimensions of the dataset. It can be interpreted from the figure that the average time taken by each test sample decreases slowly with the increase in dimensions. Similarly, Figure 6.7 depicts the scalability of the test phase of the classifier with respect to the increasing size of the dataset while keeping the dimensions constant. It is demonstrated from the figure that the average time taken by each test sample appears to increase linearly with the increase in the size of the dataset.

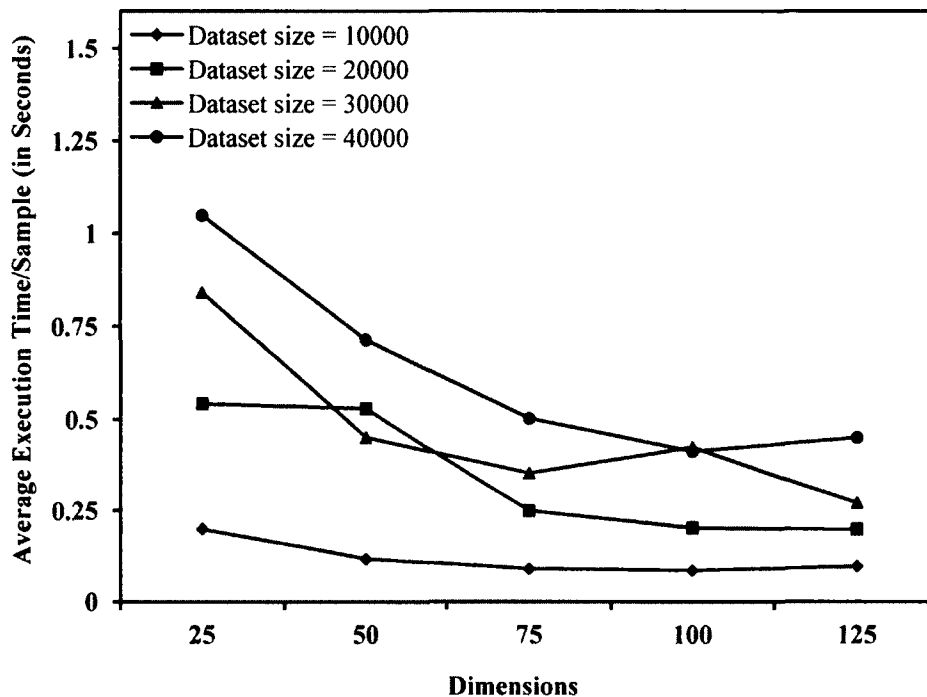


Figure 6.6: Average Execution Time/Sample v/s Dimensions

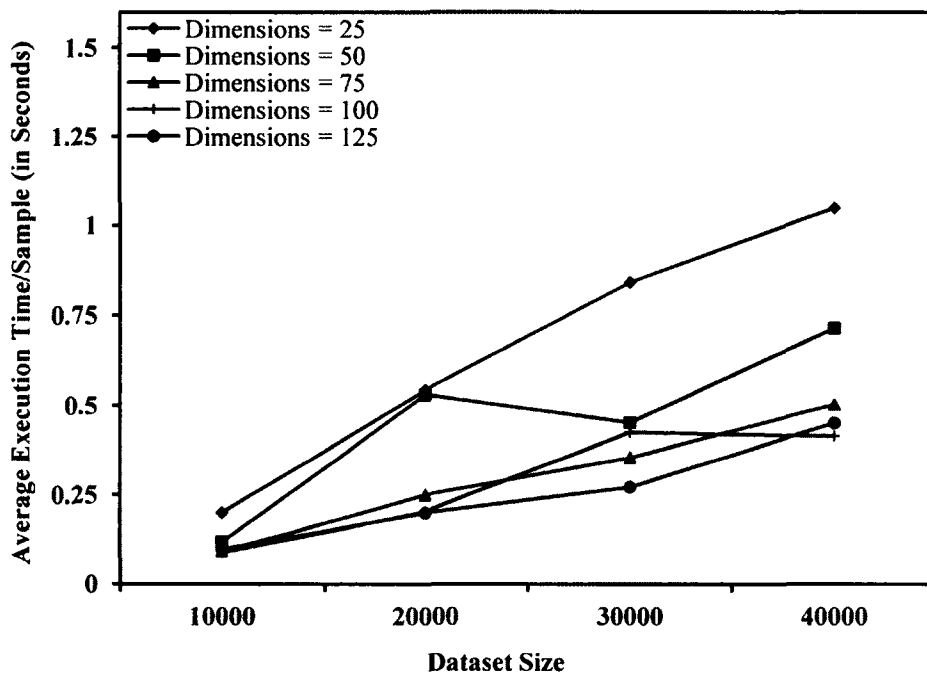


Figure 6.7: Average Execution Time/Sample v/s Dataset Size

Adaptive Grid-Based Classifier: The scalability study related to the training phase and the test phase of the classifier is as follows:

Training Phase: Figure 6.8 depicts the scalability of the training phase of the classifier with respect to the increasing dimensions of the dataset while keeping the data size constant. Figure 6.8 depicts the execution time taken by the training phase of the classifier with respect to the increasing dimensions for a given size of a dataset. It can be interpreted from the figure that the training time required by the classifier seems to increase linearly with the increase in dimensions. Similarly, Figure 6.9 depicts the scalability of the training phase of the classifier with respect to the increasing size of the dataset while keeping the dimensions constant. Figure 6.9 depicts the execution time taken by the training phase of the classifier with respect to the increasing size of the dataset for a given number of dimensions. It is demonstrated from the figure that the time required by the classifier appears to increase linearly with the increase in the size of the dataset.

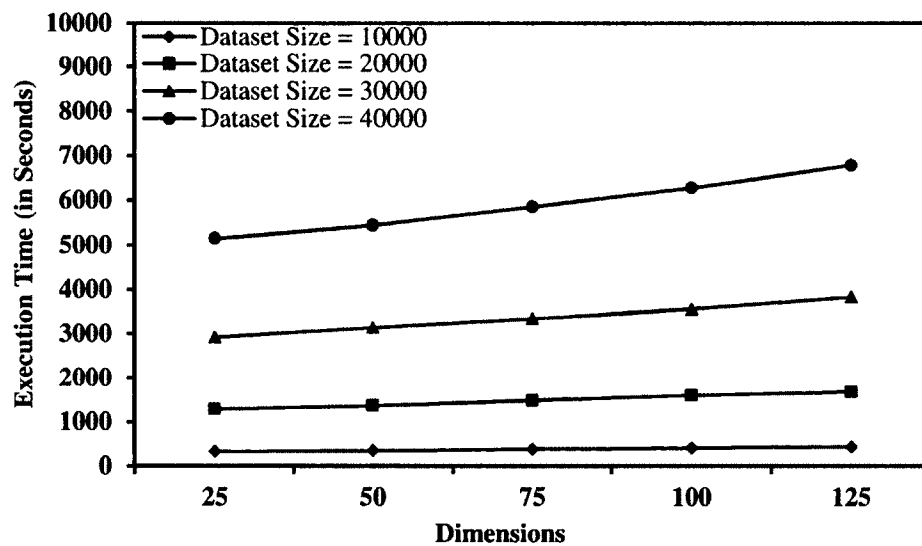


Figure 6.8: Execution Time v/s Dimensions

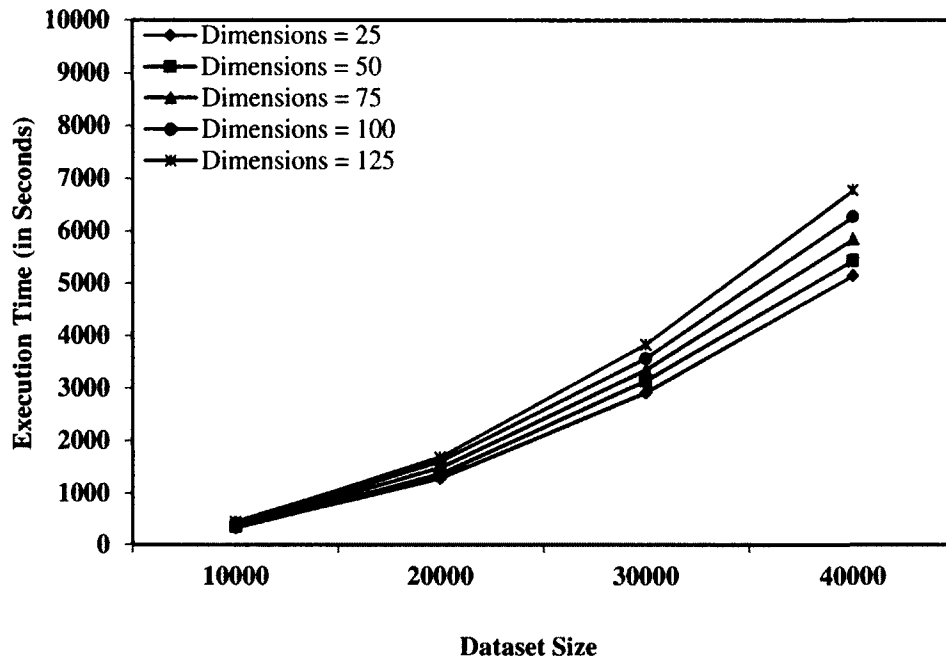


Figure 6.9: Execution Time v/s Dataset Size

Test Phase: Figure 6.10 presents the scalability of the test phase of the classifier with respect to the increasing dimensions of the dataset while keeping the data size constant, and it depicts the average execution time taken by each test sample with respect to the increasing dimensions of the dataset. It can be inferred from the figure that the average time taken by each test sample increases slowly with the increase in dimensions. Figure 6.11 depicts the scalability of the test phase of the classifier with respect to the increasing size of the dataset while keeping the dimensions constant. Figure 6.11 depicts the average execution time taken by each test sample with respect to the increasing size of the dataset. It is demonstrated from the figure that the average time taken by each test sample seems to increase linearly with the increase in the size of the dataset.

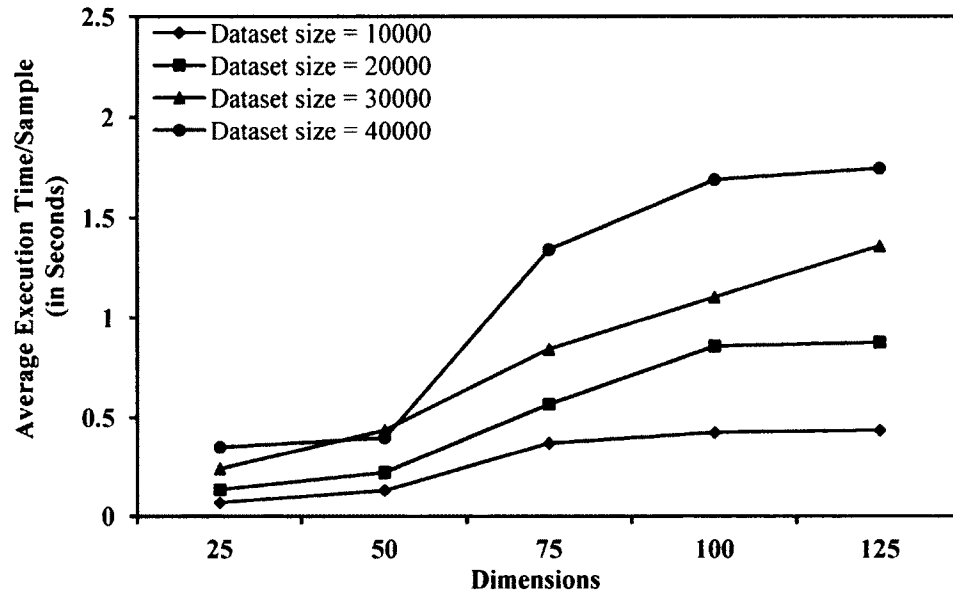


Figure 6.10: Average Execution Time/Sample v/s Dimensions

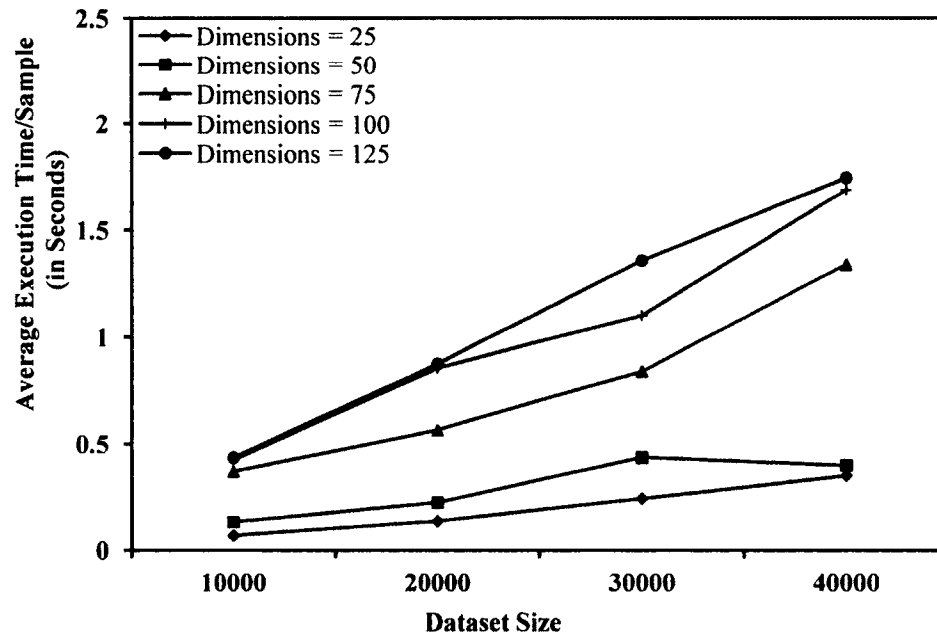


Figure 6.11: Average Execution Time/Sample v/s Dataset Size

The experimental results presented in this section demonstrate that fixed grid-based classifier and adaptive grid-based classifier are scalable and their training time and test time appear to increase linearly with the increase in data size and dimensions.

6.4.3.2 Comparative Analysis

A comparative analysis is conducted on three different datasets to demonstrate the ability of the grid-based classifiers in correctly classifying the test data. Classification results of the grid-based classifiers are compared with classification results of other well-known classifiers. The classifiers that are used for comparison include C4.5, Naïve Bayes, Classification Tree, PART rule based classifier, KNN and Logistic regression [66, 67, 71, 72]. This comparative analysis is conducted using five fold cross-validation on all the datasets. Figure 6.12 shows the classification results of the letter recognition dataset. The classification results of the fixed grid-based and adaptive grid-based classifiers are compared with C4.5, Naïve Bayes, PART, and Classification Tree. It is demonstrated in the plot that the average F-measure and classification accuracy of the grid-based classifiers are the highest and are better at correctly identifying the test samples.

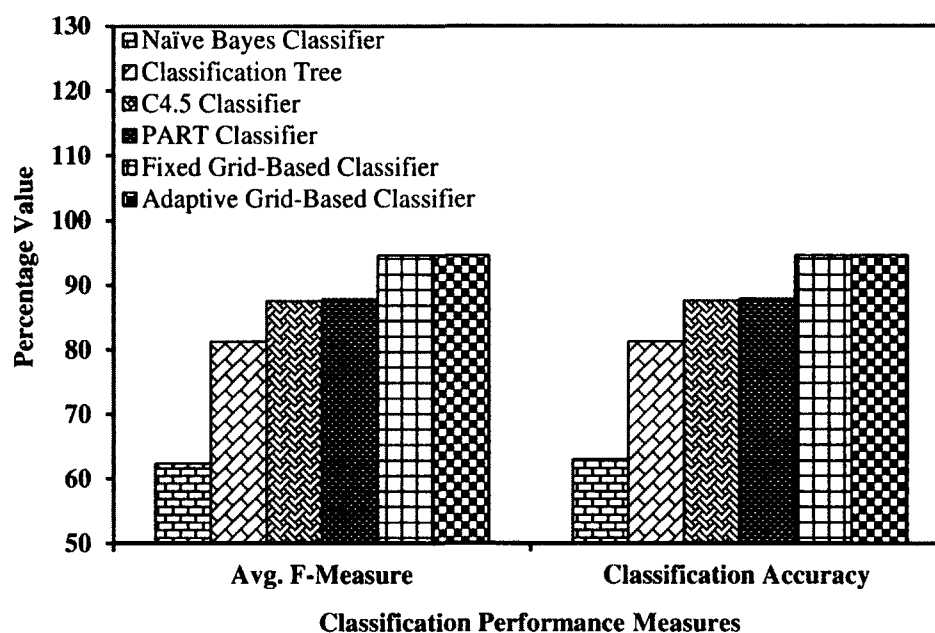


Figure 6.12: Comparative Study on Letter Recognition Dataset

Furthermore, in Figure 6.13, the classification results of the handwritten numeral recognition dataset related to the profile correlation feature set are presented. It is demonstrated from the plot that average F-measure and classification accuracy of the grid-based classifiers are superior in comparison to C4.5, Naïve Bayes, PART, and Classification Tree in correctly identifying the test data. Classification results are also compared on a protein structural classification dataset. A comparative study is presented in Figure 6.14. The study shows that the average F-measure and classification accuracy of the grid-based classifiers are better than Logistic and KNN classifiers. It is easy to interpret that grid-based classifiers demonstrate superiority over other selected classifiers.

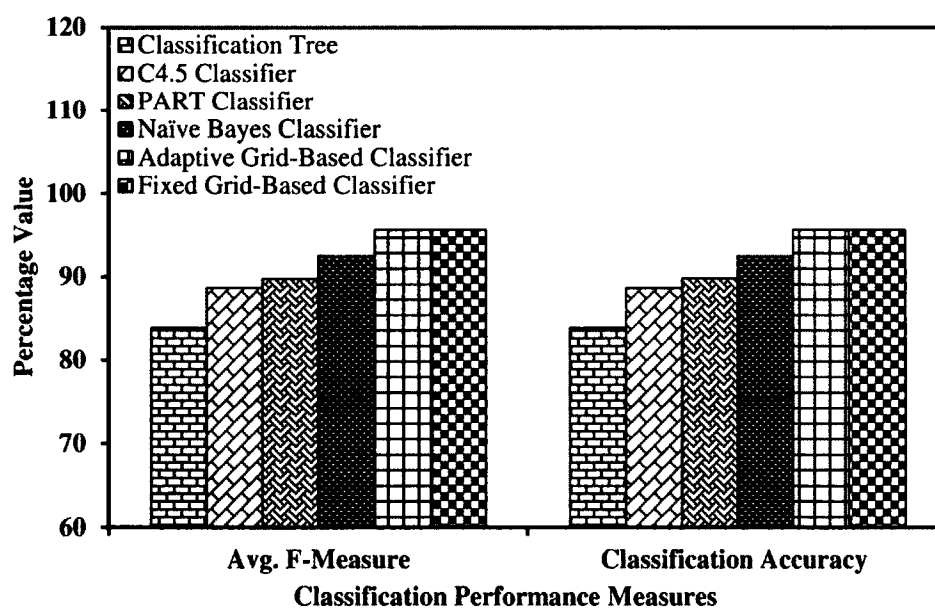


Figure 6.13: Comparative Study on Profile Correlation Feature Set

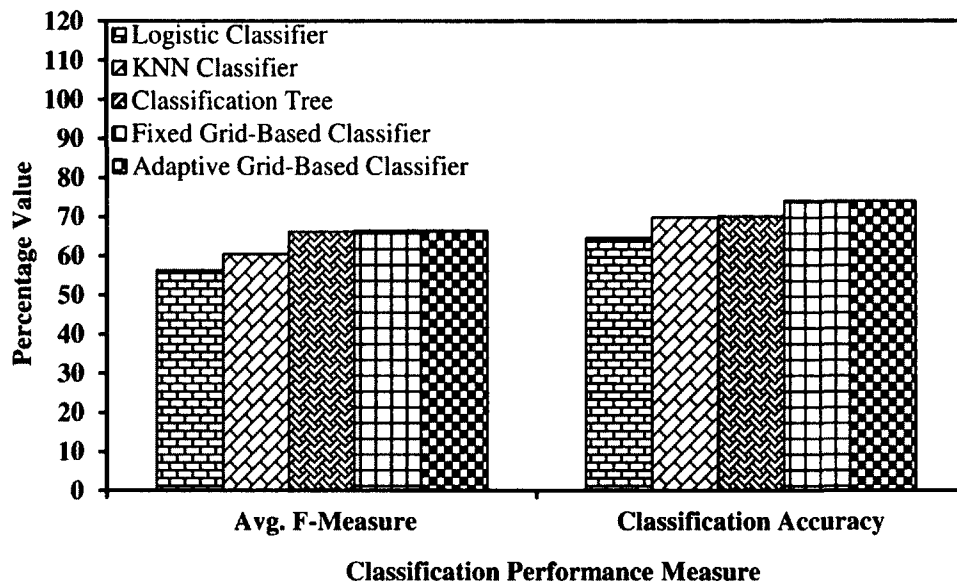


Figure 6.14: Comparative Study on Protein Structural Classification Dataset

6.4.4 Time Complexity Analysis

The time complexity analysis related to the adaptive grid generation algorithm, training phase of the classifier, and test phase of the classifier is as follows:

- 1. Grid Generation Algorithm:** In this algorithm, the total number of micro-partitions Mp_{Number} is first calculated. Next, the minimum variance based selective agglomerative hierarchical partitioning is performed, which takes $O(Mp_{Number} * (Mp_{Size})^2 + Mp_{Number})$ time. This process is repeated for all the dimensions. Thus, the time complexity of the algorithm for all dimensions is $O(d * (Mp_{Number} * (Mp_{Size})^2 + Mp_{Number}))$.
- 2. Classifier Training Phase:** In the training phase, first assign every training sample to a cell. Each assignment takes $O(d)$ time. Second, to add a new grid cell to the list of grid cells and identify the neighborhood of the new grid cell, the new grid cell is compared with all previously identified grid cells in the list, which requires N_{Total} comparisons and takes $O(d * N_{Total})$ time. Here, N_{Total} represents the total number

of grid cells after mapping all the training samples. Third, to update the list of the neighbors of all the existing grid cells that are also neighbor to the newly added grid cell requires $O(d * N_{Nbr})$ time. Here, N_{Nbr} is the number of neighboring grid cells. Thus, the overall maximum time required for the training phase can be given by $O(N * (d + dN_{Total} + dN_{Nbr}))$.

- 3. Classifier Test Phase:** In the test phase, first identify the grid cell of the test sample, which takes $O(d)$ time. Second, compare the grid cell ID of the test sample with the list of grid cell ID's of the training data. It takes $O(d * N_{Total})$. Here, N_{Total} represents the total number of grid cells after mapping all the training samples. Third, compute the distance between the test sample and the medoid of the training samples present in each neighboring grid cell. This step takes $O(d * C * N_{Nbr})$. Here, N_{Nbr} is the number of neighboring grid cells. Fourth, compute the distance between the test sample and the medoid of training samples present in N_{Total} grid cells. It takes $O(d * C * N_{Total})$. Thus, the overall time complexity of the test phase for a test sample is $O(d + d * N_{Total} + d * C * N_{Nbr} + d * C * N_{Total})$.

6.5 Conclusion

This chapter has outlined the potential of grid-based localized learning in designing fast and scalable classifier to process large datasets. Furthermore, two grid-based classification models have been developed to harness the advantage of data space partitioning. The first grid-based classification model uses uniform grid structure and the second classification model uses adaptive grid structure. The developed grid-based classification models consist of four phases: Data preprocessing phase, Grid generation phase, Training phase, and Test phase. All the phases of fixed grid-based classifier and

adaptive grid-based classifier are identical except the grid generation phase. Experiments are conducted on synthetic datasets that demonstrate that developed grid-based classifiers are scalable and demonstrate a slow and linear increase in the execution time of their training phase and test phase with an increase in the number of dimensions and size of the datasets. The comparative study conducted on real datasets has demonstrated that developed grid-based classifiers performance better than other well-known classifiers. There are still some open questions such as what would be the effect of the integration of feature ranking with the classification model for high dimensional datasets and what would be the effect of supervised data partitioning method for grid generation. These open questions regarding the grid-based classification models can be explored as future directions.

CHAPTER 7

GRID-BASED LOCALIZED LEARNING FOR CLUSTERING

Multidimensional datasets exhibit sparseness, which increases as dimensions increase [13, 14]. The sparseness of multidimensional datasets is a serious impediment to clustering algorithms and severely affects the performance of these algorithms. This problem has been addressed in the past by augmenting clustering algorithms with specialized data preprocessing techniques that reduce the overall sparseness of the data [13, 14, 20, 21, 22, 23, 24]. These data preprocessing techniques are categorized as sparseness reduction techniques and are commonly called data shrinking or data movement techniques. In such clustering techniques, first, a data shrinking algorithm is applied to diminish the sparseness of the data by moving the data points along the direction of the density gradient, which provides more condensed and demarcated clusters in the original dimensional space while retaining the dimensions [13, 14]. Clustering algorithms augmented with a data shrinking technique perform better than traditional clustering algorithms [13, 14]. However, existing data shrinking based clustering algorithms have deficiencies which need to be addressed. Therefore, there is a need to develop new algorithms that are efficient and better than existing algorithms.

The remainder of the chapter is organized as follows. In Section 7.1, research motivation is discussed. In Section 7.2, a problem statement is discussed. In Section 7.3, the methodology of the non-uniform grid-based shrinking and clustering algorithm is discussed. In Section 7.4, experimental study is presented and discussed. Finally, in Section 7.5, conclusions are presented.

7.1 Research Motivation

The deficiencies of the existing data shrinking algorithms and the deficiencies of the existing grid-based clustering algorithms have motivated us to develop a new data shrinking based clustering algorithm. The existing data shrinking algorithms suffer from the inherent instability in the shrinking process and large computational time of these algorithms [13, 14, 20, 21, 22, 23, 24]. Similarly, the existing grid-based shrinking and clustering algorithms impose uniform grid structure on all the dimensions. However, non-uniform grids are more effective than uniform grids because they capture the underlying data distribution in every dimension and are computationally more efficient than uniform grids [26]. Thus, the motivation is to develop new grid-based data shrinking and clustering algorithm to address the deficiencies of existing techniques.

7.2 Problem Statement

Uniform grid-based data partitioning imposes a uniform grid structure on the data, partitions the multidimensional space into equal size partitions, and ignores the underlying data distribution. Thus, a uniform grid fails to effectively capture the underlying data distribution in each dimension. Consequently, uniform grid-based data shrinking algorithms do not perform better than traditional algorithms. On the other hand, non-uniform grid-based data partitioning is data driven and effectively captures the

underlying data distribution in each dimension. It is highly likely that non-uniform grid-based shrinking will perform better, and it is hypothesized that non-uniform grid-based partitioning will provide effective movement of data points for data shrinking. Based on this hypothesis, the aim is to develop non-uniform grid-based data shrinking and clustering algorithm that uses non-uniform grid-based localized learning paradigm.

7.3 Methodology

The overall methodology of the developed adaptive grid-based data shrinking and clustering algorithm consists of four steps. The overall methodology is presented in Figure 7.1.

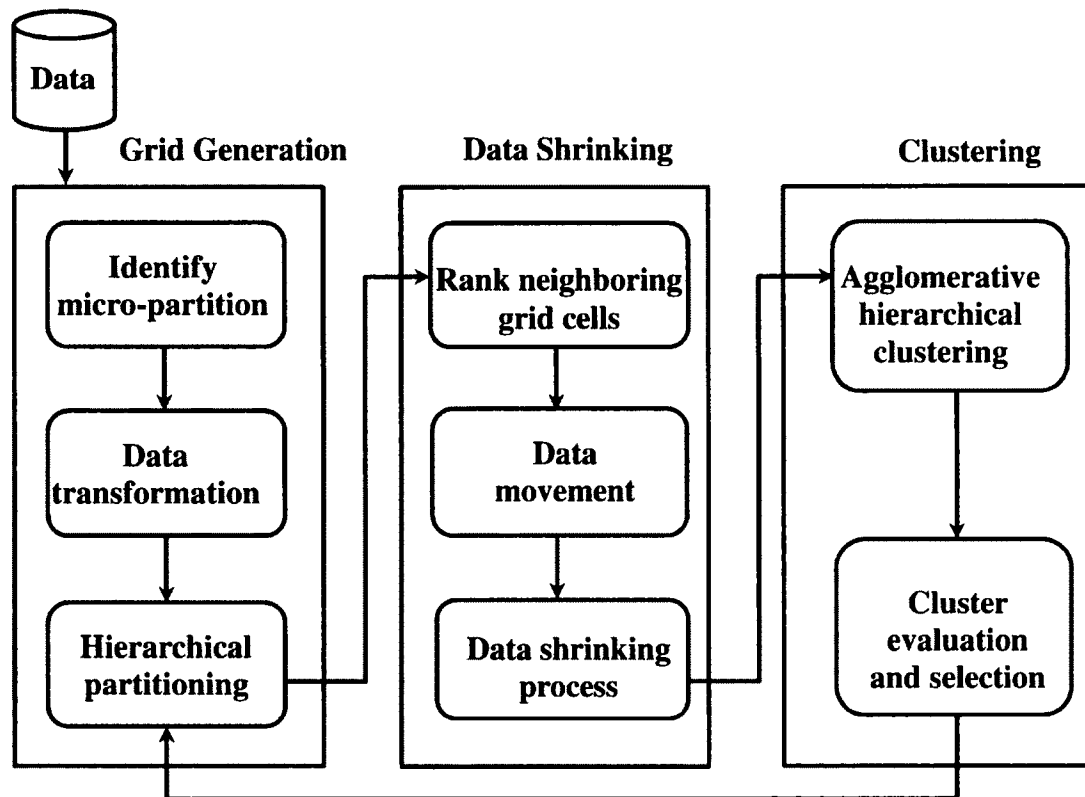


Figure 7.1: Adaptive Shrinking Based Clustering Approach

The overall methodology is divided into four sections. In Section 7.3.1, data preprocessing step is discussed. In Section 7.3.2, the non-uniform/adaptive grid generation step is discussed. In Section 7.3.3, the data shrinking step is discussed. Finally, in Section 7.3.4, the grid-based hierarchical clustering algorithm is discussed.

7.3.1 Data Preprocessing

Data preprocessing is an essential process in this methodology. In data preprocessing, the dataset is first normalized by applying Z-score normalization. Then, each dimension is transformed based on the mean and standard deviation of the dimension. The dataset is further normalized into a unit hypercube $[0, 1]^d$ to scale all the dimensions between the range of zero and one by applying min-max normalization on each dimension [2]. In addition to this step, those dimensions are eliminated from the datasets that do not provide significant variability within the dimension. The removal of the dimensions occurs when significant numbers of data values in a dimension are either zero or constant.

7.3.2 Adaptive Grid Generation

The data adaptive grid generation algorithm is a data driven technique used to create data adaptive grid structure for shrinking and clustering. This algorithm generates a data adaptive grid by creating data adaptive partitions in each dimension. The data adaptive grid is generated in three steps. Initially, micro-partitions (see Definition 3.18) are created. Next, data space transformation is performed on each micro-partition using discrete wavelet transform. Feature extraction is then performed on each transformed micro-partition by extracting a compact spectral representation and, finally, these transformed micro-partitions are clustered using a multi-objective selective

agglomerative hierarchical partitioning (MOSAH partitioning) algorithm. The following three steps are performed on each dimension.

7.3.2.1 Finding Micro-Partitions

Initially, the dimension is sorted in ascending order. The sorted one-dimensional data points are in close proximity with their neighbors. To group these points together, micro-partitions are introduced. The use of the micro-partitions is motivated by the idea that this method will reduce the computation time of the overall grid generation process. Non-overlapping units of data points called micro-partitions are created by grouping k contiguous data points ($k < N$, where N is the total number of data points). A small value of k is chosen because micro-partitions should be as small as possible but not small enough to undermine the benefits for the overall grid generation process. The choice for the size of micro-partitions is inspired by [68, 69]. In [68] and [69], $\lfloor \sqrt{N} \rfloor$ intervals are used to divide the attribute, and each interval contains approximately $\lfloor \sqrt{N} \rfloor$ intervals. The size of micro-partitions is obtained by applying Equation 7.1 and the number of micro-partitions is obtained by applying Equation 7.2:

$$Mp_{Size} = 2^{\lfloor \log_2 \lfloor \sqrt{N/10} \rfloor \rfloor}, \quad \text{Eq. 7.1}$$

$$Mp_{Number} = \left\lfloor \frac{N}{Mp_{Size}} \right\rfloor. \quad \text{Eq. 7.2}$$

In Equation 7.1, $\log_2 \sqrt{N/10}$ is used to obtain the size of micro-partitions which ensures that the micro-partitions obtained are small enough. Choosing a value smaller than $N/10$ will reduce the size of micro-partitions and will create too small micro-partitions and undermine the benefits of micro-partitioning.

7.3.2.2 Data Transformation

Data space transformation is usually applied to transform data space and obtain a new representation of the data. This process is applied to each micro-partition for extracting a compact spectral signature and data reduction. An efficient wavelet transform method called discrete wavelet transform is selected [73]. In discrete wavelet transform, wavelet coefficients are calculated only for dyadic scales and positions. Thus, the method provides more concise and efficient transformation [73]. The discrete wavelet transform of a data vector x is given by Equation 7.3, where ψ represents an impulse response called a mother wavelet. The discrete wavelet transform of a data vector x is calculated by passing it through a series of filters. The data vector x is decomposed simultaneously using both a high-pass and a low-pass filter. The output is outlined in the detail coefficients and in the approximation coefficients, respectively. The output of the transformation (detail and approximate coefficients) is given by Equations 7.4 and 7.5.

The approximate coefficients are the high-scale, low-frequency components of the data, and the detail coefficients are the low-scale, high-frequency components of the data. Approximate coefficients are more important than detail coefficients because they contain more than 98% of the energy of the data [73, 74, 75]. For these experiments, the Haar wavelet is selected as a mother wavelet because it is the simplest wavelet imaginable. Only approximate coefficients are retained after transformation in order to extract a compact spectral signature of each micro-partition:

$$W(j, k) = \sum_j \sum_k x(k) 2^{-j/2} \psi(2^{-j}n - k), \quad \text{Eq. 7.3}$$

$$y_{\text{High}}[k] = \sum_n x[n]g[2k - n], \quad \text{Eq. 7.4}$$

$$y_{\text{Low}}[k] = \sum_n x[n]h[2k - n]. \quad \text{Eq. 7.5}$$

7.3.2.3 MOSAH Partitioning

In multi-objective selective agglomerative hierarchical partitioning (MOSAH), all the transformed micro-partitions are clustered by applying a multi-objective criterion that groups contiguous micro-partitions in a bottom-up fashion until all the micro-partitions are in one cluster (see Figure 7.2).

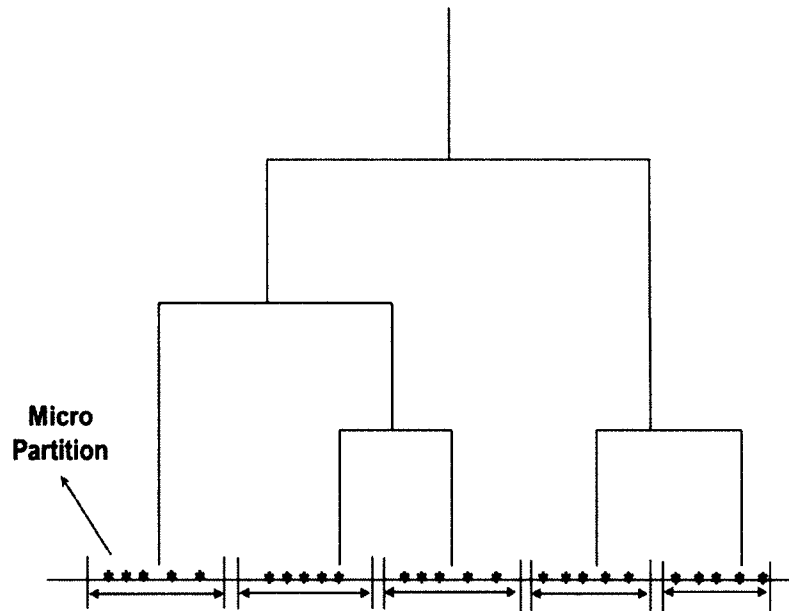


Figure 7.2: MOSAH Partitioning of Micro-partitions

In this multi-objective framework, three objective functions are used to obtain the consensus for grouping micro-partitions. These three objective functions are average-linkage, centroid-linkage, and ward-linkage. Since micro-partitions are created from a sorted dimension, all the micro-partitions are arranged in a contiguous or sequential order. The sequential order of micro-partitions also gives the MOSAH partitioning its unique characteristics. Since the micro-partitions are in sequential order, only contiguous micro-partitions are merged to form macro-partitions.

7.3.2.4 Algorithmic Description

Initially, all the transformed one-dimensional micro-partitions are given as an input to the algorithm. The pseudo-code of the algorithm is presented in Figure 7.3.

```

Algorithm: Data Adaptive Grid Generation
Input: Dataset X
Output: Data Adaptive Grid G
01  $N = \text{Number of Datapoints in } X$ 
02  $d = \text{Number of Dimensions in } X$ 
03 for  $j=1$  to  $d$ 
04    $S_d = \text{Sort}(D_j)$ 
05    $Mp_{Size} = 2^{\lceil \log_2 \lceil \sqrt{N/10} \rceil \rceil}$ 
06    $Mp_{Number} = \left\lfloor \frac{N}{Mp_{Size}} \right\rfloor$ 
07 for  $r=1$  to  $Mp_{Number}$ 
08    $m(r) = \text{Find\_micro\_partition}(D_j)$ 
09    $DWm(r) = \text{Find\_discrete\_wavelet\_transform}(m(r))$ 
10    $DWa(r) = \text{Find\_approx\_wavelet\_coeff}(DWm(r))$ 
11 end
12    $n = Mp_{Number}$  // Start of MOSAH partitioning
13 for  $r=1$  to  $(n - 1)$ 
14    $AVERAGE(r) = \frac{1}{(n_r * n_{r+1})} \sum_{i=1}^{n_r} \sum_{j=1}^{n_{r+1}} |DWa(r, i) - DWa(r + 1, j)|$ 
15    $CENTROID(r) = \frac{|DWa(r) - DWa(r + 1)|}{(n_r + n_{r+1})}$ 
16    $WARD(r) = \frac{(n_r * n_{r+1}) |DWa(r) - DWa(r+1)|}{(n_r + n_{r+1})}$ 
17 end
18 while  $n \neq 2$ 
19    $MinIndex(1) = \text{Find\_minimum}(AVERAGE)$ 
20    $MinIndex(2) = \text{Find\_minimum}(CENTROID)$ 
21    $MinIndex(3) = \text{Find\_minimum}(WARD)$ 
22    $Merge_{Index} = \text{Majority\_voting}(MinIndex)$ 
23    $M_n = \text{Merge\_micro\_partitions}(Merge_{Index})$ 
24    $n = n - 1$ 
25 end
26 for  $n=2$  to  $Mp_{Number}$ 
27    $M_n(m) = \text{Find\_partitions\_in\_original\_space}(M_n)$ 
28 end //End of MOSAH partitioning
29 end

```

Figure 7.3: Data Adaptive Grid Generation Algorithm

The algorithm begins with the computation of the proximity between all pairs of adjacent micro-partitions using the multi-objective framework to group micro-partitions. Next, two contiguous micro-partitions are grouped together to form macro-partitions based on the majority voting scheme. In this voting scheme, a pair of adjacent micro-partition is grouped together if they obtain at least two out of three votes of being the closest of all pairs of adjacent micro-partitions. The process of grouping adjacent micro-partitions continue in bottom-up fashion until all the micro-partitions are grouped together in one big partition. Once the algorithm is terminated, corresponding micro-partitions are grouped in the original data space, and a hierarchical tree of partitions is obtained in the original space.

7.3.3 Adaptive Grid-Based Shrinking

The data adaptive grid-based shrinking algorithm begins once hierarchical decomposition of data adaptive partitions is obtained for all the dimensions. For this algorithm, the user must first select a level from the hierarchical decomposition of adaptive partitions. The steps to perform data shrinking at a specified level of hierarchical decomposition are given throughout this section.

7.3.3.1 Ranking Neighboring Grid Cells

In a grid-based data movement process, the neighborhood is defined based on the grid cell [13, 14]. In general, a d -dimensional grid cell C can have $C_{Neighbor}$ distinct neighboring grid cells. There are a total of $\prod_{j=1}^d S_j - 1$ distinct neighbors. These $C_{Neighbor}$ distinct neighbors can be further categorized into d categories. The categorization of neighboring grid cells in d -dimensional data space is based on the number of facets shared between a grid cell C and its neighboring grid cells [6]. The grid

cells that share a maximum number of facets ($d - 1$) are the closest to the grid cell C . An example of ranking neighboring grid cells is shown in Figure 7.4.

		2,4		4,4
Dimension 2	1,3	2,3	3,3	4,3
		2,2		4,2
	1,1	2,1	3,1	4,1
	Dimension 1			

Figure 7.4: A Two-Dimensional Grid with Cell ID's

In Figure 7.4, a two-dimensional grid is used to demonstrate the various neighboring grid cells that can be identified in a two-dimensional grid. In the grid structure, the horizontal axis represents dimension-1, and the vertical axis represents dimension-2. Each dimension is divided into four partitions. The grid cell numbering is based on the convention, $C = (I_{1,p_1}, I_{2,p_2})$, where I_{1,p_1} represents the partition number in dimension-1 and I_{2,p_2} represents the partition number in dimension-2. A grid cell with ID (2,3) is depicted in green, and its neighboring cells are depicted in purple and orange. The orange grid cells that have the cell IDs (1,3), (3,3), (2,4), and (2,2) share one facet with the green grid cell in the center. Similarly, the purple grid cells that have the cell IDs (1,4), (3,4), (1,2), and (3,2) share no facet with the green cell in the center. Therefore, the grid cell with cell ID (2,3) has two categories of neighbors. Similarly, for higher dimensions d ($d > 2$), d types of neighbors can be identified based on the number of facets shared between the neighboring cells.

7.3.3.2 Data Movement Model

A grid-based model of attraction is employed to move all the data points in a particular grid cell as a single unit. First, identify all of grid cell C_u 's non-empty neighboring grid cells. Second, rank all neighboring grid cells using the ranking method. Third, choose all top ranked neighboring grid cells. Fourth, compute the data centroid of the selected top ranked neighboring grid cells of the grid cell C_u and the data centroid of the grid cell C_u . Fifth, move all the data points in the grid cell C_u using the data displacement formula.

To formally describe the data movement for a grid cell, let C_u be a grid cell that contains a set \mathbb{X}_u of k data points $\mathbb{X}_u = \{\vec{x}_{u1}, \dots, \dots, \vec{x}_{uk}\}$, where $\mathbb{X}_u \subset \mathbb{X}$ for which data movement is to be performed. Let $C_{NBR} = (C_{n1}, C_{n2}, \dots, \dots, C_{ni})$ be a set of selected top ranked neighboring grid cells that have $(n_1, n_2, \dots, \dots, n_i)$ number of data points. Let the data centroid of all the data points in the set C_{NBR} of grid cells be given by Equation 7.6. Similarly, the data centroid of all the points in the grid cell C_u is given by Equation 7.7:

$$\vec{c}_{NBR} = \frac{\sum_{j=1}^{ni} \left(\frac{1}{n_j} (\sum_{r=1}^{n_j} x_r) \right)}{\sum_{j=1}^{n_j} n_j}, \quad \text{Eq. 7.6}$$

$$\vec{c}_u = \frac{1}{k} (\sum_{i=1}^k x_{ui}). \quad \text{Eq. 7.7}$$

Therefore, the movement or the displacement of a data point \vec{x}_{ui} in the grid cell C_u is given by Equation 7.8:

$$\vec{x}'_{ui} = \vec{x}_{ui} + (\vec{c}_{NBR} - \vec{c}_u). \quad \text{Eq. 7.8}$$

The movement or displacement of all the other data points is performed in the grid cell C_u . The movement of data points is performed if it satisfies the movement threshold criteria given by Equation 7.9:

$$Distance(\vec{c}_{NBR}, \vec{c}_u) \geq M_{Th}. \quad \text{Eq. 7.9}$$

7.3.3.3 Data Shrinking Process

The data shrinking algorithm is multilevel data adaptive grid-based shrinking algorithm in which data shrinking is performed at each selected level. The decomposition starts at level-0 which is the root level. At this level, all the data points are in a single partition. The next level is level-1 at which the data points partition into two data adaptive partitions, and so on and so forth (see Figure 7.5). The number of levels of hierarchical decomposition is chosen such that there are fewer non-empty grid cells than data points. Once the number of levels of hierarchical decomposition is selected then data shrinking is performed at each selected level. The pseudo-code of the algorithm is presented in Figure 7.6.

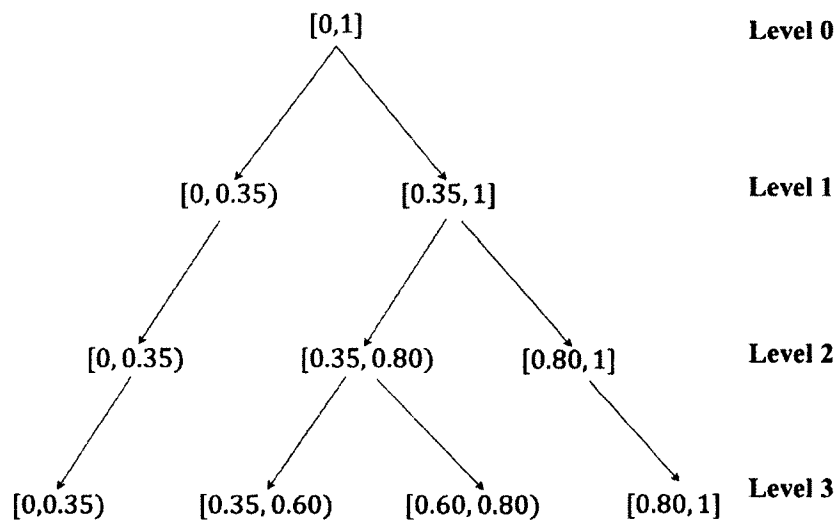


Figure 7.5: Hierarchical Decomposition of Data Adaptive Partitions

Algorithm: Data Shrinking Algorithm

Input: Grid G_1 , Dataset \mathbb{X} , Iterations l_{Th} , Threshold M_{Th}

Output: Data after Shrinking \mathbb{X}_i

```

01  $N =$  Number of Datapoints in  $\mathbb{X}$ 
02  $d =$  Number of Dimensions in  $\mathbb{X}$ 
03  $Count = 0$ 
04 for  $i=1$  to  $N$ 
05    $C(i) =$  Find_Cell_Id( $\vec{X}_i, G_1$ )
06   if ( $C(i) \notin Z$ ) then
07     Add  $C(i)$  to  $Z$ 
08      $Count = Count + 1$ 
09     Add  $X_i$  to  $Zdata(Count).data$ 
10   end
11 for  $m=1$  to  $Count$ 
12   if ( $Z(m) == C(i)$ ) then
13      $Z(m).count = Z(m).count + 1$ 
14     Add  $X_i$  to  $Zdata(m).data$ 
15   end
16 end
17 end
18  $l = 0$ 
19 while  $l \leq l_{Th}$ 
20   [ $Zs, Zsdata$ ] = Sort( $Z, Zdata$ )
21    $n = 1$ 
22   while  $n \leq Count$ 
23     Find Neighboring Cells of Cell  $Zs(n)$ 
24     Compute Centroid  $\vec{c}_{NBR}$  of Neighboring Cells
25     Compute Centroid  $\vec{c}_n$  of Cell  $Zs(n)$ 
26     if ( $Distance(\vec{c}_{NBR}, \vec{c}_n) \geq M_{Th}$ ) then
27       Compute Displacement of Datapoints in  $Zsdata(n)$ 
28     end
29   end
30   if(No Movement between  $l$  and  $l + 1$ ) then
31     Exit
32   end
33 end

```

Figure 7.6: Pseudo-code for Data Shrinking Algorithm

The algorithm first maps all the data points on the adaptive grid. During this process, it identifies all non-empty grid cells and corresponding data points, and accumulates all the data points that are mapped to the non-empty grid cells. The algorithm then sorts all non-empty grid cells in increasing order based on the number of data points in them. Grid cells are sorted in increasing order of the number of data points

to insure that the data points in sparse regions are processed first and then moved toward denser regions. In this step, all the grid cells are arranged in an order. As a result, the proposed shrinking algorithm is insensitive towards the order of the input data points and it does not suffer from this deficiency like other existing algorithms. Then, the first grid cell is taken from the sorted list of cells, and its neighboring grid cells are identified to select the top ranked neighboring cells. Once the neighboring grid cells are identified, the data points in the grid cell are moved according to the model of data movement and are reassigned to new grid cells. This process is repeated for all the grid cells in the sorted list. After the movement of the data points, all empty grid cells are removed from the list and all non-empty grid cells are kept. This process is repeated for the specified number of iterations or until the data points no longer move in any two contiguous iterations.

7.3.4 Adaptive Grid-Based Clustering

The developed clustering algorithm is a grid-based hierarchical clustering algorithm in which each grid cell is considered a single unit. See Figure 7.7 for a graphical representation of the algorithm.

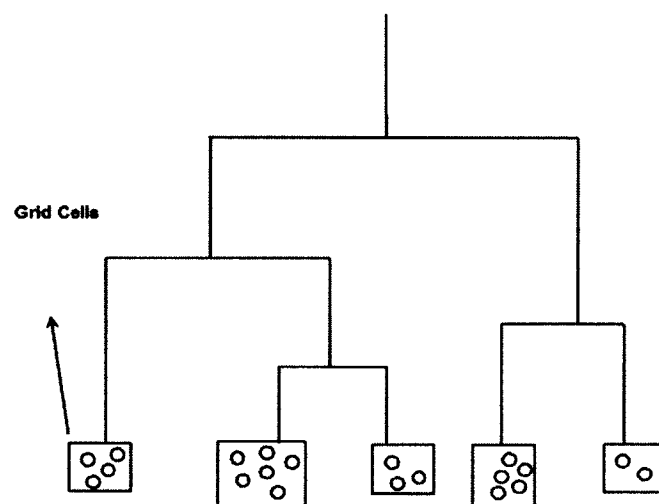


Figure 7.7: Grid-Based Hierarchical Clustering

Therefore, a multi-objective voting scheme is used in the hierarchical clustering of nonempty grid cells. The voting scheme uses average linkage, centroid linkage, and ward linkage measures for the clustering. Once data shrinking is performed for the selected level of hierarchical decomposition, data is passed to the clustering algorithm. A pseudo-code of the algorithm is presented in Figure 7.8.

Algorithm: Clustering Algorithm
Input: Nonempty cells Cells, Nonempty cells data CellsData
Output: Hierarchical Clusters

```

01 Nc = Nonempty Cells
02 Clusters = Nc
03 Step = 1
04 for j=1 to Clusters
05   AVERAGE(Clusterr, Clusterj) =  $\frac{1}{(n_r + n_j)} \sum_{i=1}^{n_r} \sum_{k=1}^{n_j} |\vec{x}_{i,k} - \vec{x}_{i,k}|$ 
06   CENTROID(Clusterr, Clusterj) =  $|\vec{c}_r - \vec{c}_j|$ 
07   WARD(Clusterr, Clusterj) =  $(n_r * n_j) \frac{|\vec{c}_r - \vec{c}_j|}{(n_r + n_j)}$ 
08 end
09 while Clusters ≠ 1
10   RMSSTD(Step, 1) = Compute_RMSSTD(CellsDataStep)
11   MinIndex(1) = Find_minimum(AVERAGE)
12   MinIndex(2) = Find_minimum(CENTROID)
13   MinIndex(3) = Find_minimum(WARD)
14   Indexr,j = Majority_voting(MinIndex)
15   Merge_Clusters(Indexr,j)
16   Clusters = Clusters - 1
17   Step = Step + 1
18 end

```

Figure 7.8: Pseudo-code for Adaptive Grid-Based Clustering

The algorithm begins with the computation of the proximity between all pairs of data centroids of nonempty grid cells based on the proposed multi-objective framework that uses average linkage, centroid linkage, and ward linkage criterion to cluster nonempty grid cells. Two nonempty grid cells are clustered based on the majority voting scheme. In the proposed voting scheme, a pair of nonempty grid cells is grouped if they

obtain at least two of three votes from three linkage criterion for being the closest pair of nonempty grid cells. This process of clustering nonempty grid cells continues until all nonempty grid cells are grouped into one cluster and form a hierarchical tree.

7.4 Results and Discussion

This section presents all the experimental studies and discussions related to the developed algorithm. In this section, discussion about the time complexity analysis, clustering evaluation and validation, scalability study, and a comparative study is presented.

7.4.1 Datasets

Both real and synthetic datasets with a wide range of dimensions and sample size are used for experiments and to assess the capabilities of the developed clustering algorithm. A detailed description of each of these datasets is as follows.

- 1. Wine Recognition Dataset:** The first dataset is the Wine Recognition dataset. This dataset has 13 dimensions and 178 data points. The dataset contains three clusters and each cluster contains 59, 71, and 48 data points, respectively. The dataset is available at the UCI machine learning repository [58].
- 2. Ecoli Dataset:** The second dataset is the Ecoli dataset, which pertains to protein localization site data. This dataset has 7 dimensions and 336 data points. The dataset contains 8 clusters and each cluster has 143, 77, 52, 35, 20, 5, 2, and 2 data points, respectively. The dataset is available at the UCI machine learning repository [58].
- 3. Protein Structural Classification Dataset:** The protein dataset consists of feature vectors that are based on amino acid sequence of corresponding proteins. The feature construction is based amino acid composition, physical and stereo chemical

properties of amino acids. Each feature vector consists of 125 feature descriptor. The dataset has 582 samples and is divided into 5 protein structural classes namely α , β , $\alpha + \beta$, $\frac{\alpha}{\beta}$, and *Small* proteins. The feature vector construction method is discussed in [60]. This data is available at (<http://ranger.uta.edu/~chqing/protein/>).

- 4. Synthetic Dataset:** A set of synthetic datasets is used for the comparative analysis of the algorithms. Three synthetic datasets are generated with 50, 60, and 120 dimensions and four clusters each. Datasets are generated randomly using the separation index of 0.1 which indicates that these generated clusters are close to each other [70]. The dataset with 50 dimensions contains 2049 data points in four clusters (c1=414, c2=566, c3=652, c4=417). The dataset with 60 dimensions contains 2017 data points in four clusters (c1=515, c2=496, c3=549, c4=457) and the dataset with 120 dimensions contains 2062 data points in four clusters (c1=543, c2=580, c3=522, c4=417). Another set of synthetic datasets is also generated for the scalability analysis of the algorithms. A set of datasets with 10, 20, 30, 40, and 50 dimensions, and 2,000, 4000, 6000, 8000, and 10,000 data points is generated. Each dataset contains two clusters, each of which has an equal number of data points in respective datasets. Two clusters are generated from a normal distribution with means of 10, -10 and a standard of deviation 3.

7.4.2 Validation

In this clustering method, clusters are obtained as hierarchical decomposition of the data points. The root-mean-square standard deviation (RMSSTD) measure is used to obtain the optimal number of clusters from the hierarchical decomposition, which is represented by Equation 7.10. The root-mean-square standard deviation (RMSSTD)

measures the compactness or homogeneity of clusters formed at a given level of hierarchical decomposition [76, 77, 78, 79]. A small value of RMSSTD indicates the clusters formed at a given level are formed by merging two homogeneous clusters and a large value of RMSSTD indicates that the clusters formed at a given level are formed by merging two heterogeneous clusters. The optimal number of clusters is obtained by employing the 'Elbow/ method' [79].

$$RMSSTD = \sqrt{\frac{\sum_{i=1}^{n_c} \sum_{j=1}^d \sum_{k=1}^{n_{ij}} (x_k - \bar{x}_j)^2}{\sum_{i=1}^{n_c} \sum_{j=1}^d (n_{ij} - 1)}}. \quad \text{Eq. 7.10}$$

The developed clustering algorithm is compared with other clustering algorithms using external clustering validation measures precision, recall and F-measure, which are represented by Equations 7.11, 7.12, and 7.13. In Equations 7.11 and 7.12, the original clusters are represented by c_i^o , detected clusters are represented by c_i^s , and i represents the i^{th} cluster:

$$Precision = \frac{|c_i^s \cap c_i^o|}{|c_i^s|}, \quad \text{Eq. 7.11}$$

$$Recall = \frac{|c_i^s \cap c_i^o|}{|c_i^o|}, \quad \text{Eq. 7.12}$$

$$F - measure = 2 \times \left(\frac{precision \cdot recall}{precision + recall} \right). \quad \text{Eq. 7.13}$$

In the above equations, TP, TN, FP, and FN refer to true positive, true negative, false positive, and false negative, respectively.

7.4.3 Experiments

In this section, an experimental study is presented. These experiments are conducted to demonstrate the effect of sparseness with increasing dimensions and the advantages of using non-uniform grid over uniform grid.

7.4.3.1 Scalability Analysis

The scalability of grid generation, data shrinking, and clustering algorithm is demonstrated by experimenting with synthetic datasets. For experiments, five iterations are maintained for all the datasets. The scalability study of the grid generation algorithm is presented in Figures 7.9 and 7.10.

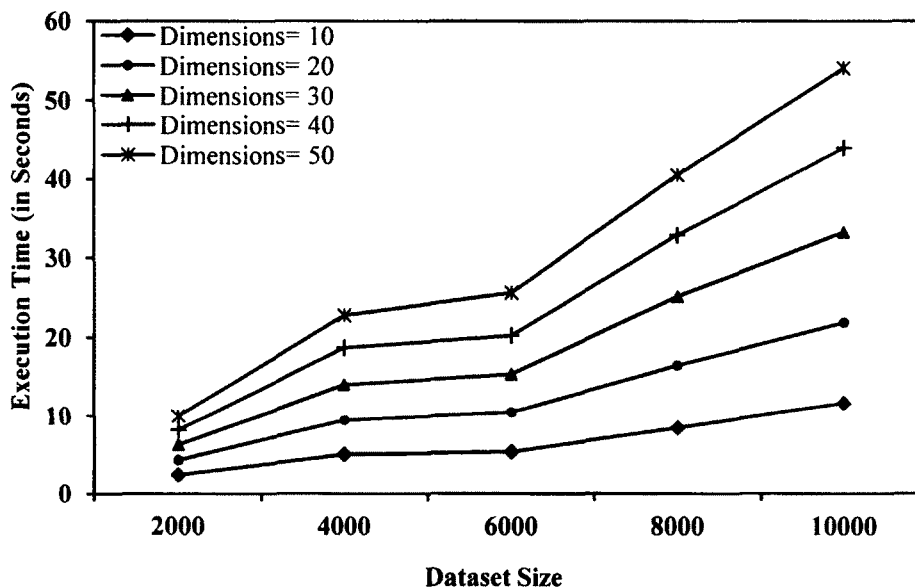


Figure 7.9: Execution Time v/s Dataset Size (Analysis for Grid Generation Method)

Figure 7.9 presents the scalability plot, showing the execution time with the increasing dataset size, and Figure 7.10 plots the execution time with the increasing number of dimensions. A grid generation algorithm is applied on each dataset to obtain a hierarchical decomposition of data adaptive partitions. It is observed from Figures 7.9 and 7.10 that the execution time of the grid generation algorithm appears to increase linearly with the increase in dataset size and dimensions.

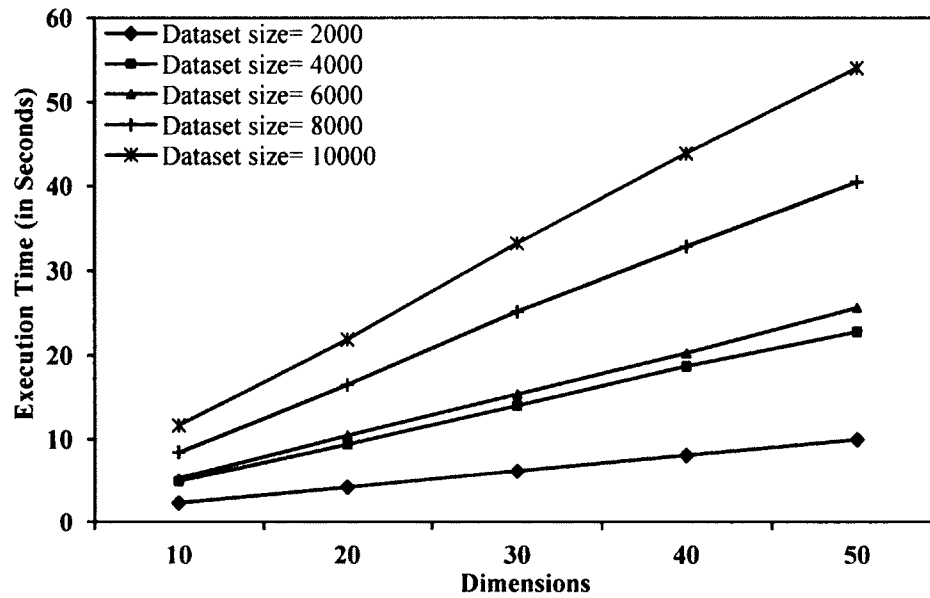


Figure 7.10: Execution Time v/s Dimensions (Analysis for Grid Generation Method)

Similarly, the scalability study related to the data shrinking algorithm is presented in which the execution time of the algorithm is studied with respect to the increasing dataset size and the increasing number of dimensions. The data shrinking algorithm is applied on each synthetic dataset, using hierarchical decomposition level – 2 for the experiments. Minimum movement threshold value M_{Th} ranging from 0.10 to 0.4 with the increments of 0.1 is used. For every dataset, the average execution time over all M_{Th} values is plotted. It is observed in Figures 7.11 and 7.12 that the execution time of the data shrinking algorithm appears to increase non linearly with respect to the dataset size and linearly with respect to the dimensions.

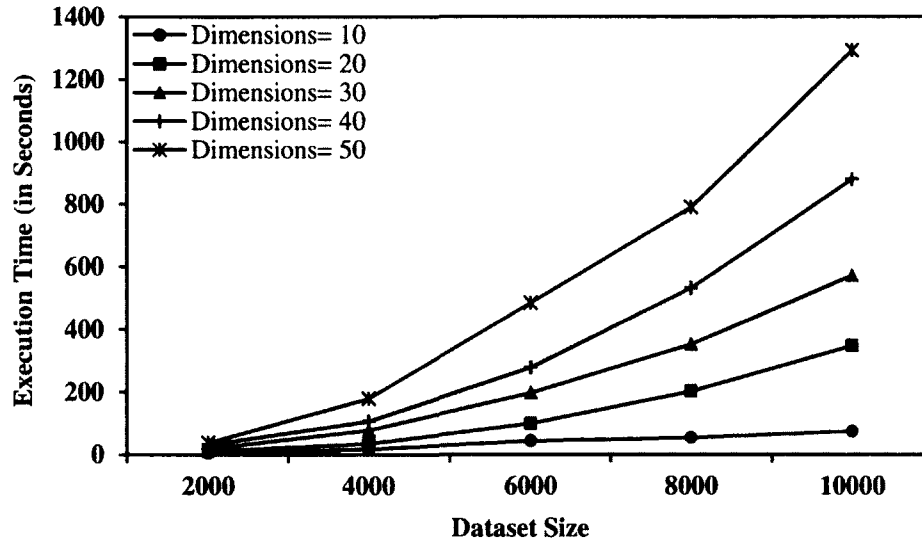


Figure 7.11: Execution Time v/s Dataset Size (Analysis for Data Shrinking Method)

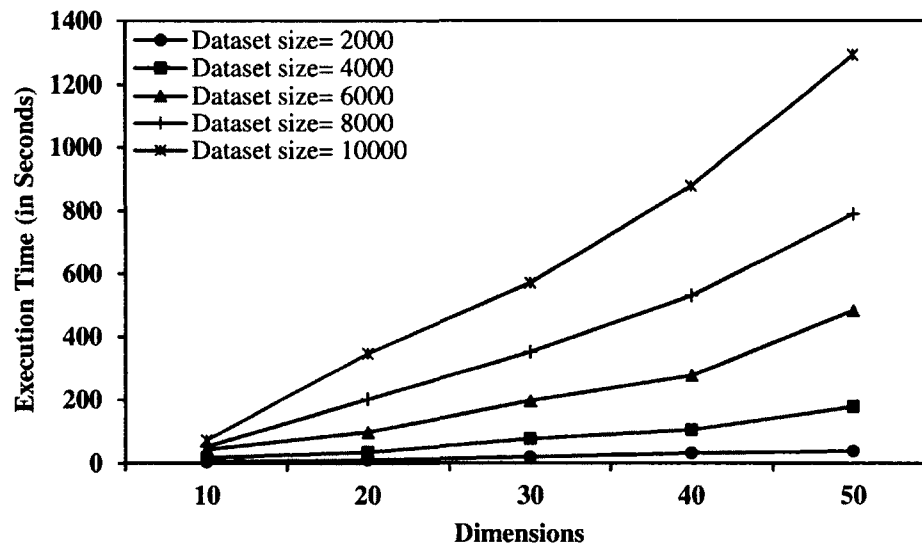


Figure 7.12: Execution Time v/s Dimensions (Analysis for Data Shrinking Method)

Finally, the scalability study of the adaptive grid-based clustering algorithm is presented in Figures 7.13 and 7.14. Figure 7.13 presents the scalability plot, showing the execution time of the algorithm with an increasing dataset size, and Figure 7.14 shows the execution time of the algorithm with an increasing number of dimensions.

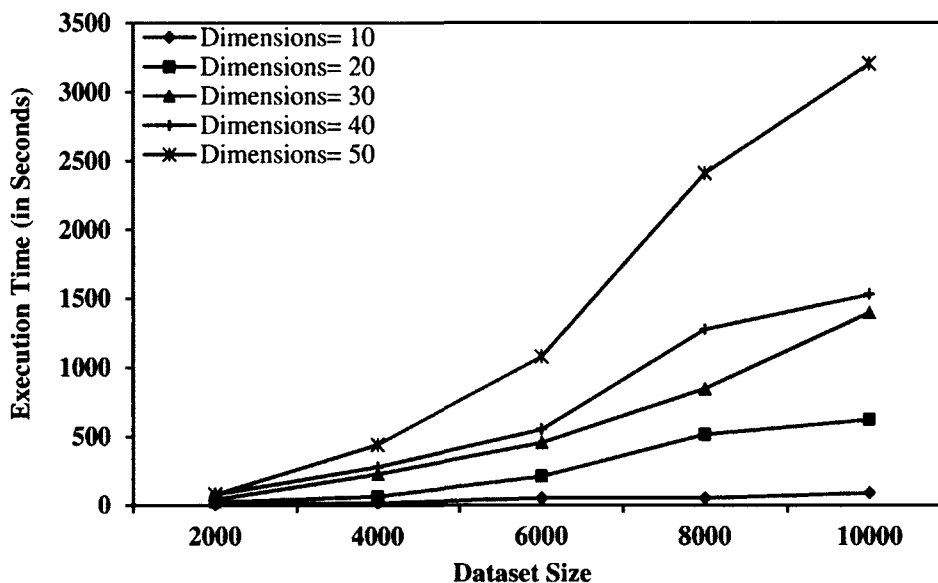


Figure 7.13: Execution Time v/s Dataset Size (Analysis for Clustering Method)

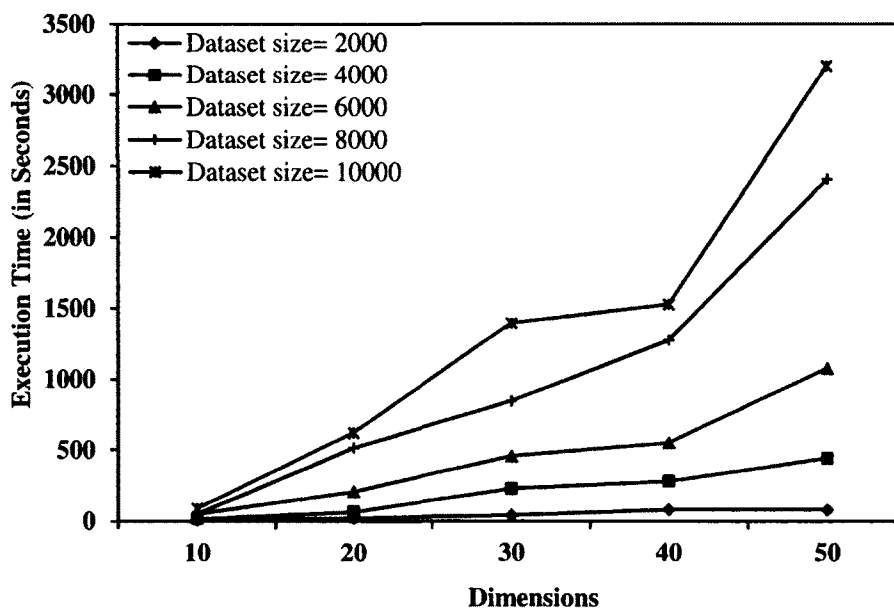


Figure 7.14: Execution Time v/s Dimensions (Analysis for Clustering Method)

The hierarchical clustering algorithm is applied on each synthetic dataset after the data shrinking algorithm and the average execution time of the clustering algorithm over all the M_{Th} value is computed. It can be observed in Figures 7.13 and 7.14 that the

average execution time of the clustering algorithm appears to increase non linearly with the increase in dataset size and dimensions.

7.4.3.2 Comparative Analysis

A set of experiments is conducted to evaluate the clustering algorithm.

Experiments are also conducted to compare the developed algorithm with the uniform grid algorithm method and other clustering algorithms such as CURE and DBSCAN [13, 52, 53]. A brief description of control parameters for these algorithms is as follows:

CURE requires three input parameter options: $-k$ for the number of clusters, $-\alpha$ for the shrinking factor of CURE, and $-r$ for the number of representative points of the cluster.

DBSCAN requires two input parameter options: Eps- a neighborhood distance and MinPts- the minimum number of data points in an Eps neighborhood.

1. Experiments on Wine Recognition Dataset: Experiments on this dataset demonstrate that the proposed algorithm performs better than the benchmark method and other clustering algorithms. A data adaptive grid-based clustering algorithm is applied on the dataset. The minimum movement threshold M_{Th} value is set at a range of 0.10 to 0.35 with increments of 0.025. Experiments are run to obtain clusters for a combination of (M_{Th}, level) . Then RMSSTD is used to identify the number of clusters using the 'Elbow method'. The results of the adaptive shrinking based clustering and the benchmark algorithm are compared in Table 7.1. The F-measure is used to compare the overall performance of the two algorithms. The average F-measure between the two methods (benchmark approach=89.69% and adaptive shrinking based approach=93.75%) indicates that the adaptive shrinking based method achieves an overall better performance than the benchmark method.

Table 7.1: Benchmark v/s Adaptive Shrinking Based Method on Wine Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	59	53	53	100.00	89.83	94.64
	Proposed	59	64	57	89.06	96.61	92.68
i=2	Benchmark	71	52	52	98.08	71.83	82.93
	Proposed	71	65	63	96.92	88.73	92.64
i=3	Benchmark	48	46	43	93.48	89.58	91.49
	Proposed	48	50	47	94.00	97.92	95.92

A comparative study is also conducted between the CURE clustering algorithm and the adaptive shrinking based clustering algorithm. These results are presented in Table 7.2. Clustering results of CURE were obtained from [13]. The comparison indicates that the adaptive shrinking based clustering algorithm performs a better cluster detection than the CURE clustering algorithm. Next, the DBSCAN algorithm is applied on the Wine Recognition dataset. Experiments are performed by setting Eps-parameter to values ranging from 0.10 to 0.90 with increments of 0.1, and setting the MinPts parameter to values ranging from one to ten with increments of one. A comparison of the DBSCAN algorithm and the adaptive shrinking based clustering are presented the Table 7.3. The comparison of the results obtained from both the algorithms indicates that the adaptive shrinking based clustering performs better cluster detection than the DBSCAN algorithm.

Table 7.2: CURE v/s Adaptive Shrinking Based Method on Wine Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	59	72	54	75.00	91.52	82.44
	Proposed	59	64	57	89.06	96.61	92.68
i=2	CURE	71	50	41	82.00	57.77	67.78
	Proposed	71	65	63	96.92	88.73	92.64
i=3	CURE	48	46	26	56.52	54.16	55.32
	Proposed	48	50	47	94.00	97.92	95.92

Table 7.3: DBSCAN v/s Adaptive Shrinking Based Method on Wine Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	59	103	58	56.31	98.31	71.61
	Proposed	59	64	57	89.06	96.61	92.68
i=2	DBSCAN	71	2	2	100.00	2.82	5.49
	Proposed	71	65	63	96.92	88.73	92.64
i=3	DBSCAN	48	51	46	90.20	95.83	92.93
	Proposed	48	50	47	94.00	97.92	95.92

2. Experiments on the Ecoli Dataset: The data adaptive grid-based clustering algorithm is applied on the Ecoli dataset. The minimum movement threshold M_{Th} value is set at a range of 0.10 to 0.35 with the increments of 0.025. Once clusters are obtained for all the combinations of $(M_{Th}, level)$, RMSSTD is computed to identify the number of clusters using the 'Elbow method'. The results of the adaptive shrinking based clustering are compared with the results of the benchmark method in Table 7.4, which compares the F-measure for the overall performance of the two algorithms. The Ecoli dataset contains eight clusters, but three clusters are insignificant and contain only 5, 2, and 2 data points.

Thus, only five clusters are included in the discussion. After comparing the average F-measure between the two methods (benchmark approach =70.07% and adaptive shrinking based approach=77.82%), the adaptive shrinking based algorithm achieves an overall 8% better performance than the benchmark algorithm.

Table 7.4: Benchmark v/s Adaptive Shrinking Based Method on Ecoli Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	143	135	130	96.30	90.91	93.53
	Proposed	143	158	143	90.51	100.00	95.02
i=2	Benchmark	77	22	22	100.00	28.57	44.44
	Proposed	77	38	36	92.11	45.45	60.87
i=3	Benchmark	52	68	43	63.24	82.69	71.67
	Proposed	52	52	44	84.62	84.62	84.62
i=4	Benchmark	35	49	32	65.31	91.43	76.19
	Proposed	35	67	31	46.27	88.57	60.79
i=5	Benchmark	20	11	10	90.91	50.00	64.52
	Proposed	20	21	18	85.71	90.00	87.80

A comparative study was also conducted between the CURE clustering algorithm and the adaptive grid-based clustering algorithm on the Ecoli dataset which is presented in Table 7.5. The clustering results of CURE were obtained from [13]. The comparative study of the F-measure indicates that the adaptive shrinking based clustering algorithm performs better than the CURE clustering algorithm.

Table 7.5: CURE v/s Adaptive Shrinking Based Method on Ecoli Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	143	120	115	95.83	80.41	87.45
	Proposed	143	158	143	90.51	100.00	95.02
i=2	CURE	77	67	41	61.19	53.24	56.94
	Proposed	77	38	36	92.11	45.45	60.87
i=3	CURE	52	32	30	93.75	57.69	71.43
	Proposed	52	52	44	84.62	84.62	84.62
i=4	CURE	35	NA	NA	NA	NA	NA
	Proposed	35	67	31	46.27	88.57	60.79
i=5	CURE	20	NA	NA	NA	NA	NA
	Proposed	20	21	18	85.71	90.00	87.80

Next, the DBSCAN algorithm is applied to the Ecoli dataset and its results are presented in Table 7.6. The experiments on the DBSCAN algorithm are conducted for different parameter configurations. The Eps parameter is set to values ranging from 0.10 to 0.30 with increments of .001 and the MinPts parameter to values ranging from one to 30 with increments of one. In Table 7.6, The best clustering results obtained from the DBSCAN algorithm are compared with the results of adaptive shrinking based clustering algorithm. The comparison of precision, recall, and F-measure values corresponding to each cluster obtained from both the algorithms indicates that the adaptive shrinking based clustering algorithm performs a better cluster detection than the DBSCAN clustering algorithm on this dataset.

Table 7.6: DBSCAN v/s Adaptive Shrinking Based Method on Ecoli Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	143	124	119	95.97	83.22	89.14
	Proposed	143	158	143	90.51	100.00	95.02
i=2	DBSCAN	77	59	39	66.10	50.65	57.35
	Proposed	77	38	36	92.11	45.45	60.87
i=3	DBSCAN	52	24	22	91.67	42.31	57.90
	Proposed	52	52	44	84.62	84.62	84.62
i=4	DBSCAN	35	10	8	80.00	22.86	35.56
	Proposed	35	67	31	46.27	88.57	60.79
i=5	DBSCAN	20	NA	NA	NA	NA	NA
	Proposed	20	21	18	85.71	90.00	87.80

3. Experiments on Protein Datasets: Initially, experiments with adaptive shrinking based clustering algorithm are performed. The minimum movement threshold M_{Th} values are set ranging from 0.10 to 2.0 with the increments of 0.025 and hierarchy level of one, two and three. Once hierarchical clusters are obtained for all the combinations of (M_{Th}, level) , then the final clusters are selected. Similarly, experiments are conducted with the benchmark method. Experiments are conducted by setting the minimum movement threshold M_{Th} values ranging from 0.5 to 3.0 with the increments of 0.05 and different grid scales for cluster detection [1]. A comparison of the results obtained from both the algorithms is shown in Table 7.7, which compares the precision, recall, and F-measure for the overall performance of the two algorithms. The average F-measure between the two methods (benchmark approach=37.82% and adaptive shrinking based approach=60.13%) indicates that the adaptive shrinking based method achieves overall better cluster detection than the benchmark method.

Table 7.7: Benchmark v/s Adaptive Shrinking Based Method on Protein Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	112	NA	NA	NA	NA	NA
	Proposed	112	64	62	96.88	55.36	70.46
i=2	Benchmark	177	3	3	100.00	1.69	3.32
	Proposed	177	55	48	87.27	27.12	41.38
i=3	Benchmark	203	232	178	76.72	87.68	81.83
	Proposed	203	279	180	64.52	88.67	74.69
i=4	Benchmark	46	17	17	100.00	36.96	53.97
	Proposed	46	137	30	21.90	65.22	32.79
i=5	Benchmark	44	24	17	70.83	38.64	50.00
	Proposed	44	47	37	78.72	84.10	81.32

A comparative study is also conducted between the CURE clustering algorithm and the adaptive shrinking based clustering algorithm on this synthetic dataset, which is presented in Table 7.8. The clustering results of CURE were obtained by experimenting with different parameter settings [52]. Experiments are conducted by setting the α -parameter to values ranging from .10 to .30 with increments of .05 and the MinPts parameter to values ranging from 10 to 60 with increments of five. The comparison of the average F-measure between the two methods (CURE clustering=32.19% and Adaptive shrinking based approach=60.13%) indicates that adaptive shrinking based clustering algorithm achieves better cluster detection than the CURE clustering algorithm.

Table 7.8: CURE v/s Adaptive Shrinking Based Method on Protein Dataset

Cluster no.	Algorithm	c_i^o	c_i^f	$ c_i^f \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	112	42	38	90.48	33.93	49.35
	Proposed	112	64	62	96.88	55.36	70.46
i=2	CURE	177	206	65	31.55	36.72	33.94
	Proposed	177	55	48	87.27	27.12	41.38
i=3	CURE	203	308	188	61.04	92.61	73.58
	Proposed	203	279	180	64.52	88.67	74.69
i=4	CURE	46	3	1	33.33	2.17	4.08
	Proposed	46	137	30	21.90	65.22	32.79
i=5	CURE	44	NA	NA	NA	NA	NA
	Proposed	44	47	37	78.72	84.10	81.32

Next, the DBSCAN algorithm is applied on this synthetic dataset and its results are presented in Table 7.9. The Eps parameter is set to values ranging from .10 to 1.0 with increments of .1 and the MinPts parameter is set to values ranging from one to ten with increments of one to find the best clustering result for the DBSCAN algorithm. The comparison of the average F-measure between the two methods (DBSCAN clustering=25.13% and Adaptive shrinking based approach=60.13%) indicates that adaptive shrinking based clustering algorithm outperforms the DBSCAN clustering algorithm.

Table 7.9: DBSCAN v/s Adaptive Shrinking Based Method on Protein Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	112	25	25	100.00	22.32	36.49
	Proposed	112	64	62	96.88	55.36	70.46
i=2	DBSCAN	177	6	6	100.00	3.39	6.56
	Proposed	177	55	48	87.27	27.12	41.38
i=3	DBSCAN	203	305	189	61.97	93.10	74.41
	Proposed	203	279	180	64.52	88.67	74.69
i=4	DBSCAN	46	3	2	66.67	4.35	8.17
	Proposed	46	137	30	21.90	65.22	32.79
i=5	DBSCAN	44	NA	NA	NA	NA	NA
	Proposed	44	47	37	78.72	84.10	81.32

4. Experiments on Synthetic Datasets: A comparative analysis is also conducted on a set of synthetic dataset. The set of synthetic dataset contains data set with 50, 60 and 120 dimensions. The experiments pertaining to these three datasets are as follows:

1. Synthetic dataset with 50 dimensions: Initially, experiments are conducted with adaptive shrinking based clustering algorithm. Experiments are conducted by setting the minimum movement threshold M_{Th} values ranging from 0.10 to 1.0 with the increments of 0.1 and hierarchy level of one, two and three. Once hierarchical clusters are obtained for all the combinations of (M_{Th}, level) , then best clusters are selected. Similarly, experiments are conducted with the benchmark method, the minimum movement threshold M_{Th} -parameter is set to values ranging from 0.5 to 3.5 with the increments of 0.1 and different scales for cluster detection [1]. A comparison of results obtained from both the algorithms is shown in Table 7.10, which indicates that adaptive shrinking based clustering has a better cluster detection than the benchmark method.

Table 7.10: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	414	NA	NA	NA	NA	NA
	Proposed	414	364	354	97.25	85.51	91.03
i=2	Benchmark	566	993	566	57.00	100	72.61
	Proposed	566	541	522	96.49	92.23	94.31
i=3	Benchmark	652	19	19	100	2.91	5.66
	Proposed	652	624	585	93.75	89.72	91.69
i=4	Benchmark	417	10	9	90	2.16	4.22
	Proposed	417	520	410	78.85	98.32	87.51

A comparative study is also conducted between the CURE clustering algorithm and the adaptive shrinking based clustering algorithm on this synthetic dataset, which is presented in Table 7.11. The clustering results of CURE were obtained by experimenting with different parameter settings [52]. Experiments are conducted by setting the α -parameter to values ranging from .10 to .30 with increments of .05 and the MinPts parameter to values ranging from 10 to 60 with increments of five. The comparison presented in Table 7.11 indicates that our clustering algorithm achieves a much better cluster detection than the CURE clustering algorithm. Next, the DBSCAN clustering algorithm is applied on the dataset. Experiments are conducted by setting the Eps parameter to values ranging from .10 to 1.0 with increments of .1 and the MinPts parameter to values ranging from one to ten with increments of one and finding the best clustering result for the DBSCAN algorithm, which are presented in Table 7.12. It can be observed that adaptive shrinking based clustering outperforms the DBSCAN algorithm.

Table 7.11: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	414	NA	NA	NA	NA	NA
	Proposed	414	364	354	97.25	85.51	91.03
i=2	CURE	566	NA	NA	NA	NA	NA
	Proposed	566	541	522	96.49	92.23	94.31
i=3	CURE	652	1365	550	40.30	84.36	54.37
	Proposed	652	624	585	93.75	89.72	91.69
i=4	CURE	417	684	415	60.67	99.52	75.39
	Proposed	417	520	410	78.85	98.32	87.51

Table 7.12: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	414	60	59	98.33	14.25	24.89
	Proposed	414	364	354	97.25	85.51	91.03
i=2	DBSCAN	566	359	210	58.50	37.10	45.41
	Proposed	566	541	522	96.49	92.23	94.31
i=3	DBSCAN	652	5	5	100	0.77	1.53
	Proposed	652	624	585	93.75	89.72	91.69
i=4	DBSCAN	417	NA	NA	NA	NA	NA
	Proposed	417	520	410	78.85	98.32	87.51

2. Synthetic dataset with 60 dimensions: Initially, adaptive shrinking based clustering algorithm is applied on the dataset and the minimum movement threshold M_{Th} parameter is set to values ranging from 0.10 to 1.0 with the increments of 0.1 and hierarchy level of one, two and three. Once clusters are obtained for all the combinations of (M_{Th}, level) , then best clusters are selected. Similarly, experiments are conducted with

the benchmark method. For this method, the minimum movement threshold M_{Th} -parameter is set to values ranging from 0.5 to 3.5 with the increments of 0.1 and different grid scales are used for cluster detection [1]. The results of adaptive shrinking based clustering are compared with the results of the benchmark method in Table 7.13. After comparing the results between the two methods, the results indicate that adaptive shrinking based clustering shows a much better performance than the benchmark method.

Table 7.13: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	515	9	9	100	1.75	3.44
	Proposed	515	630	498	79.05	96.70	86.99
i=2	Benchmark	496	1401	496	35.40	100	52.29
	Proposed	496	439	404	92.03	81.45	86.42
i=3	Benchmark	549	15	15	100	2.73	5.32
	Proposed	549	538	524	97.40	95.45	96.41
i=4	Benchmark	457	9	9	100	1.97	3.87
	Proposed	457	410	395	96.34	86.43	91.11

A comparative study is also conducted between the CURE clustering algorithm and the adaptive shrinking based clustering algorithm, which is presented in Table 7.14. The experiments are conducted for different parameter settings of the CURE clustering algorithm[52]. The α - parameter is set to values ranging from 0.10 to 0.30 with increments of .05 and the MinPts parameter is set to values ranging from 10 to 60 with increments of 5. The comparison indicates that the adaptive shrinking base clustering achieves better cluster detection than the CURE clustering algorithm.

Table 7.14: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	515	1344	471	35.04	91.46	50.67
	Proposed	515	630	498	79.05	96.70	86.99
i=2	CURE	496	NA	NA	NA	NA	NA
	Proposed	496	439	404	92.03	81.45	86.42
i=3	CURE	549	673	526	78.16	95.81	86.09
	Proposed	549	538	524	97.40	95.45	96.41
i=4	CURE	457	NA	NA	NA	NA	NA
	Proposed	457	410	395	96.34	86.43	91.11

Next, the DBSCAN algorithm is applied on this synthetic dataset and its results are presented in Table 7.15. The Eps parameter takes the values ranging from .10 to 1.0 with increments of 0.1 and the MinPts parameter takes the values ranging from one to ten with increments of one to find the best clustering result for the DBSCAN algorithm. The comparison of the results from the DBSCAN clustering algorithm and the adaptive shrinking based clustering algorithm are presented in Table 7.15. These result demonstrate that on this dataset adaptive shrinking based clustering performs better cluster detection than the DBSCAN clustering algorithm.

Table 7.15: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	515	1230	447	36.34	86.80	51.23
	Proposed	515	630	498	79.05	96.70	86.99
i=2	DBSCAN	496	2	2	100	0.40	0.80
	Proposed	496	439	404	92.03	81.45	86.42
i=3	DBSCAN	549	NA	NA	NA	NA	NA
	Proposed	549	538	524	97.40	95.45	96.41
i=4	DBSCAN	457	2	2	100	0.44	0.88
	Proposed	457	410	395	96.34	86.43	91.11

3. Synthetic dataset with 120 dimensions: The adaptive shrinking based clustering algorithm is applied by setting the minimum movement threshold M_{Th} values ranging from 0.10 to 2.0 with the increments of 0.1 and hierarchy level of one, two and three. After obtaining the clusters for all the combinations of (M_{Th}, level) , best clusters are selected. Similarly, experiments are conducted with the benchmark method. The minimum movement threshold M_{Th} -parameter is set to values ranging from 0.5 to 4.0 with the increments of 0.1 and different grid scales are used for cluster detection [1]. The Table 7.16 shows the comparison of both the clustering method, which compares the precision, recall, and F-measure for the two algorithms. After comparing the two methods, it is evident that adaptive shrinking based clustering performs better than the benchmark method.

Table 7.16: Benchmark Method v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	Benchmark	543	30	30	100	5.52	10.46
	Proposed	543	624	505	80.93	93.00	86.55
i=2	Benchmark	580	973	580	59.61	100	74.67
	Proposed	580	511	469	91.78	80.86	85.97
i=3	Benchmark	522	34	34	100	6.51	12.22
	Proposed	522	525	465	88.57	89.08	88.82
i=4	Benchmark	417	34	34	100	8.15	15.07
	Proposed	417	402	344	85.57	82.49	84.00

Next, a comparative study is conducted between the CURE clustering algorithm and adaptive shrinking based clustering algorithm for this dataset, which is presented in Table 7.17. Experiments on the CURE clustering algorithm are conducted for different parameter configuration [52]. The α - parameter is set to values ranging from .10 to .30 with increments of .05 and the MinPts parameter is set to values ranging from 10 to 60 with increments of five. The comparison presented in Table 7.17 demonstrates that the adaptive shrinking based clustering has a better cluster detection than the CURE clustering algorithm. Finally, the DBSCAN algorithm is used for the experiments. To find the best clustering result, the Eps parameter is set to values ranging from 0.50 to 1.5 with increments of .1 and the MinPts parameter is set to values ranging from one to 10 with increments of one. The results presented in Table 7.18 conclude that the adaptive shrinking based clustering algorithm has better performance than the DBSCAN algorithm.

Table 7.17: CURE v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	CURE	543	686	524	76.38	96.50	85.27
	Proposed	543	624	505	80.93	93.00	86.55
i=2	CURE	580	1376	507	36.85	87.41	51.84
	Proposed	580	511	469	91.78	80.86	85.97
i=3	CURE	522	NA	NA	NA	NA	NA
	Proposed	522	525	465	88.57	89.08	88.82
i=4	CURE	417	NA	NA	NA	NA	NA
	Proposed	417	402	344	85.57	82.49	84.00

Table 7.18: DBSCAN v/s Adaptive Shrinking Based Method on a Synthetic Dataset

Cluster no.	Algorithm	c_i^o	c_i^s	$ c_i^s \cap c_i^o $	Precision (%)	Recall (%)	F-measure (%)
i=1	DBSCAN	543	2	2	100	0.37	0.7
	Proposed	543	510	479	93.92	88.21	86.55
i=2	DBSCAN	580	1454	531	36.52	91.55	52.21
	Proposed	580	233	212	90.99	36.55	85.97
i=3	DBSCAN	522	NA	NA	NA	NA	NA
	Proposed	522	1150	501	43.57	95.98	88.82
i=4	DBSCAN	417	NA	NA	NA	NA	NA
	Proposed	417	169	157	92.90	37.65	84.00

7.4.4 Time Complexity Analysis

The time complexity analysis related to the grid generation algorithm, data shrinking algorithm, and clustering algorithm are explained in the following list.

1. **Grid Generation Algorithm:** In this algorithm, the total number of micro-partitions Mp_{Number} is first calculated. Next, discrete wavelet transform is computed

for every micro-partition. This step takes $O(Mp_{Size})$ time. Thus, the time complexity of the overall process is $O(Mp_{Number} * Mp_{Size})$. Next, multi-objective selective agglomerative hierarchical partitioning is performed. This step takes $O(Mp_{Number} * (Mp_{Size})^2)$ time. Thus, the time complexity of the algorithm for all dimensions is $O(d * Mp_{Number} * (Mp_{Size})^2)$.

2. **Data Shrinking Algorithm:** In this algorithm, first, map N data points on a grid structure and then find all nonempty cells N_{Cells} . This step takes $O(d * N * N_{Cells})$ time. Next, perform shrinking which takes $O(N_{Cells}^2)$ time for a single iteration. Therefore, the overall time complexity of the algorithm for I iterations is $O(I * N_{Cells}^2)$.
3. **Clustering Algorithm:** The clustering algorithm is a grid-based hierarchical clustering algorithm in which nonempty grid cell are clustered in agglomerative fashion. If N_{Cells} represent the number of nonempty grid cells, then the time complexity of the algorithm is $O(N_{Cells}^2)$.

7.5 Conclusion

In this chapter, a new shrinking based clustering algorithm is presented. The developed algorithm is an adaptive grid-based data shrinking and clustering algorithm that addresses the limitations of existing data shrinking based clustering algorithms. Three unique algorithms have been explained in this chapter: a multi-objective selective agglomerative hierarchical partitioning algorithm to generate multilevel adaptive grids, an adaptive grid-based data shrinking algorithm to reduce the sparseness of the multidimensional datasets, and a grid-based hierarchical clustering algorithm to detect clusters. Experimental results have demonstrated that the developed algorithm can

produce superior and competitive results when compared with other shrinking based clustering algorithms and traditional clustering algorithms.

CHAPTER 8

CONCLUSIONS

The research presented in this dissertation is aimed to develop novel learning techniques for data mining and addressing the important issues such as data sparseness, high dimensionality, and large size of the datasets. Application of the grid-based localized learning paradigm was envisaged to achieve this goal. As a result, supervised and unsupervised learning methods are developed that utilize grid-based localized learning paradigm [80, 81]. The details of the contribution of this dissertation are discussed in the following sections.

8.1 Contribution to Grid-Based Supervised Learning

In this dissertation, two methods are presented related to the supervised learning. The first method is a feature ranking method. It is based on the unique data shrinking profile of each feature, which is computed after performing the data shrinking operation. It is based on the hypothesis that every dimension that participates in the shrinking process shrinks in a unique way and can be used to find the most discriminating features. The experimental results also confirm the hypothesis. The second method is a classification algorithm. It utilizes the grid-based learning paradigm for the classification model. The classification models consist of the data preprocessing phase, the grid generation phase, the training phase and the test phase. The experimental study also indicates that grid-based classifiers are scalable and demonstrate a linear increase in the

execution time with an increase in the number of dimensions and size of the datasets. These two methods provide a unique contribution in the area of supervised learning and dimensionality reduction [80, 81].

8.2 Contribution to Grid-Based Unsupervised Learning

In this dissertation, a clustering algorithm is presented which is related to the unsupervised learning paradigm. A novel approach of shrinking based clustering is presented that aims to address the limitations of the existing data shrinking approaches by utilizing the adaptive grid structures for data shrinking and clustering. It is based on the hypothesis that adaptive grid structures are more effective than uniform grid structures. The experimental study also confirms the hypothesis. This method provides a unique contribution in the area of unsupervised learning and sparseness reduction methods.

The experimental studies have established the potential of adaptive grid-based localized learning for both supervised and unsupervised computational frameworks. The contribution of the above mentioned novel algorithms not only lays the foundation for research in this direction, but it also opens new venues for research in this direction. In this realm of data mining, there are still lots of open questions and opportunities that can be explored. As a future direction, these algorithms can be further enhanced by focusing on improving their computational time and memory space requirements. Similarly, these algorithms can be utilized for handling massive datasets by parallelizing these algorithms.

REFERENCES

- [1] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview," *AI Magazine*, vol. 17, no. 3, pp. 37-54, 1996.
- [2] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," Second Edition, San Francisco: Morgan Kaufmann Publishers, 2006.
- [3] U. Fayyad, G. Piatetsky-shapiro and P. Smyth, "Knowledge Discovery and Data Mining: Towards a Unifying Framework," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, 1996.
- [4] E. Alpaydin, "Introduction to Machine Learning," Second Edition, Cambridge: The MIT Press, 2010.
- [5] T. Hastie, R. Tibshirani and J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," Second Edition, New York: Springer, 2009.
- [6] E. Schikuta, "Grid Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets," in *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, 1996.
- [7] A. Hinneburg and D. A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, New York, 1998.
- [8] G. Sheikholeslami, S. Chatterjee and A. Zhang, "WaveCluster: A Wavelet Based Clustering Approach for Spatial Data in Very Large Databases," *The VLDB Journal*, vol. 8, no. 3-4, pp. 289-304, 2000.
- [9] Y. Sun and Y. Lu, "A Scalable Grid-Based Clustering Algorithm for Very Large Spatial Databases," in *Proceedings of the 2006 International Conference on Computational Intelligence and Security*, Guangzhou, 2006.
- [10] C. Xiaoyun, M. Yufang, Z. Yan and W. Ping, "GMDBSCAN: Multi-Density DBSCAN Cluster Based on Grid," in *Proceedings of the 2008 IEEE International Conference on E-Business Engineering*, Xi'an, 2008.

- [11] F. Angiulli, C. Pizzuti and M. Ruffolo, "DESCRY: A Density-Based Clustering Algorithm for Very Large Data sets," in *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning*, Exeter, 2004.
- [12] J. T. Rickard, R. R. Yager and W. Miller, "Mountain Clustering on Non-uniform Grids," in *Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop*, Washington, DC, 2004.
- [13] Y. Shi and A. Zhang, "A Shrinking Based Dimension Reduction Approach for Multi-Dimensional Data Analysis," in *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, Santorini Island, 2004.
- [14] Y. Shi, Y. Song and A. Zhang, "A Shrinking Based Approach for Multidimensional Data Analysis," in *Proceedings of the 29th International Conference on Very Large Data Bases*, Berlin, 2003.
- [15] P. Kontkanen and P. Myllymäki, "MDL Histogram Density Estimation," in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, San Juan, 2007.
- [16] A. Hanselmann, O. C. Schrempf and U. D. Hanebeck, "Optimal Parametric Density Estimation by Minimizing an Analytic Distance Measure," in *Proceedings of the 10th International Conference on Information Fusion*, Québec, 2007.
- [17] A. Elgammal, R. Duraiswami and L. S. Davis, "Efficient Kernel Density Estimation Using the Fast Gauss Transform with Applications to Color Modeling and Tracking," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1499-1504, 2003.
- [18] S. R. Sain, "Multivariate Locally Adaptive Density Estimation," *Computational Statistics & Data Analysis*, vol. 39, no. 2, pp. 165-186, 2002.
- [19] X. Wei, H. Huang and S. Tian, "A Grid-Based Clustering Algorithm for Network Anomaly Detection," in *Proceedings of the 1st International Symposium on Data, Privacy and E-Commerce*, Chengdu, 2007.
- [20] S. Kundu, "Gravitational Clustering: A New Approach Based on the Spatial Distribution of the Points," *Pattern Recognition*, vol. 32, no. 7, pp. 1149-1160, 1999.

- [21] T. V. Ravi and K. C. Gowda, "Clustering of Symbolic Objects Using Gravitational Approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 6, pp. 888-894, 1999.
- [22] J. Gomez, D. Dasgupta and O. Nasraoui, "A New Gravitational Clustering Algorithm," in *Proceedings of the 3rd SIAM International Conference on Data Mining*, San Francisco, 2003.
- [23] C. Cariou, K. Chehdi and A. Nagle, "Gravitational Transform for Data Clustering - Application to Multicomponent Image Classification," in *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, 2005.
- [24] X. Wang, Q. Weiliang and R. H. Zamar, "CLUES: A Nonparametric Clustering Method Based on Local Shrinking," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 286-298, 2007.
- [25] A. Hinneburg and D. A. Keim, "Optimal Grid Clustering: Towards Breaking the Curse of Dimensionality in High Dimensional Clustering," in *Proceedings of the 25th International Conference on Very Large Data Bases*, Edinburgh, 1999.
- [26] H. Nagesh, S. Goil and A. Choudhary, "Adaptive Grids for Clustering Massive Data Sets," in *Proceedings of the 1st SIAM International Conference on Data Mining*, Chicago, 2001.
- [27] B. L. Milenova and M. M. Campos, "O-Cluster: Scalable Clustering of Large High Dimensional Data Sets," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, Maebashi City, 2002.
- [28] J.-p. Zhang, Y. Yang, J. Yang, Z.-b. Zhang and Z. Liu, "Spatial Clustering Algorithm Based on Optimized-Division," in *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery*, Haikou, 2007.
- [29] M. Glomba and U. Markowska-Kaczmar, "IBUSCA: A Grid-Based Bottom-up Subspace Clustering Algorithm," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications*, Jinan, 2006.
- [30] Y. Shi, "A Dimension Reduction Approach Using Shrinking for Multi-Dimensional Data Analysis," *International Journal of Intelligent Information Processing*, vol. 1, no. 2, pp. 86-98, 2010.

- [31] A. K. Cherukuri, "Analysis of Unsupervised Dimensionality Reduction Techniques," *Computer Science and Information Systems*, vol. 6, no. 2, pp. 217-227, 2009.
- [32] M. Dash, H. Liu and J. Yao, "Dimensionality Reduction of Unsupervised Data," in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, Newport Beach, 1997.
- [33] C. Bartenhagen, H.-U. Klein, C. Ruckert, X. Jiang and M. Dugas, "Comparative Study of Unsupervised Dimension Reduction Techniques for the Visualization of Microarray Gene Expression Data," *BMC Bioinformatics*, vol. 11, no. 11, pp. 320-330, 2010.
- [34] J. Choo, H. Kim, H. Park and H. Zha, "A Comparison of Unsupervised Dimension Reduction Algorithms for Classification," in *Proceedings of the 2007 IEEE International Conference on Bioinformatics and Biomedicine*, Silicon Valley, 2007.
- [35] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *The Journal of Machine Learning Research*, vol. 3, no. 1, pp. 1157-1182, 2003.
- [36] P. Mitra, C. A. Murthy and S. K. Pal, "Unsupervised Feature Selection Using Feature Similarity," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301-312, 2002.
- [37] J. G. Dy and C. E. Brodley, "Feature Selection for Unsupervised Learning," *The Journal of Machine Learning Research*, vol. 5, no. 1, pp. 845-889, 2004.
- [38] H. Liu and L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Transaction on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491-502, 2005.
- [39] Y. Li, B.-L. Lu and Z.-F. Wu, "A Hybrid Method of Unsupervised Feature Selection Based on Ranking," in *Proceedings of the 18th International Conference on Pattern Recognition*, Hong Kong, 2006.
- [40] H.-L. Wei and S. A. Billings, "Feature Subset Selection and Ranking for Data Dimensionality Reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 162-166, 2007.
- [41] J. Novakovic, P. Strbac and D. Bulatovic, "Toward Optimal Feature Selection Using Ranking Methods and Classification Algorithms," *Yugoslav Journal of Operations Research*, vol. 21, no. 1, pp. 119-135, 2011.

- [42] W. Duch, T. Wiczczyński, J. Biesiada and M. Blachnik, "Comparison of Feature Ranking Methods Based on Information Entropy," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, Budapest, 2004.
- [43] L. Yu and H. Liu, "Feature Selection for High dimensional Data: A Fast Correlation Based Filter Solution," in *Proceedings of the 20th International Conference on Machine Learning*, Washington, DC , 2003.
- [44] K. Jong, J. Mary, A. Cornuejols, E. Marchiori and M. Sebag, "Ensemble Feature Ranking," in *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Pisa, 2004.
- [45] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica (Slovenia)*, vol. 31, no. 3, pp. 249-268, 2007.
- [46] A. Alzghoul and M. Löfstrand, "Increasing Availability of Industrial Systems Through Data Stream Mining," *Computers & Industrial Engineering*, vol. 60, no. 2, pp. 195-205, 2011.
- [47] T. Yeh, J. J. Lee and T. Darrell, "Scalable Classifiers for Internet Vision Tasks," in *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Anchorage, 2008.
- [48] M. Mehta, R. Agrawal and J. Rissanen, "SLIQ: A Fast Scalable Classifier for Data Mining," in *Proceedings of the 5th International Conference on Extending Database Technology*, Avignon, 1996.
- [49] J. C. Shafer, R. Agrawal and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," in *Proceedings of the 22nd International Conference on Very Large Data Bases*, Mumbai, 1996.
- [50] M. Joshi, G. Karypis and V. Kumar, "ScalParC: A New Scalable and Efficient Parallel Classifier Algorithm for Mining Large Datasets," in *Proceedings of the 12th IEEE International Parallel Processing Symposium*, Orlando, 1998.
- [51] S. Guha, R. Rastogi and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, 1998.

- [52] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, 1996.
- [53] M. Ankerst, M. M. Breunig and H.-P. Kriegel, "OPTICS: Ordering Points To Identify the Clustering Structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia, 1999.
- [54] G. Karypis, E.-H. Han and V. Kumar, "Chameleon: Hierarchical Clustering Using Dynamic Modeling," *Computer*, vol. 32, no. 8, pp. 68-75, 1999.
- [55] S. Guha, R. Rastogi and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," in *Proceedings of the 15th International Conference on Data Engineering*, Sydney, 1999.
- [56] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, 1996.
- [57] A. Frank and A. Asuncion, "UCI Machine Learning Repository," University of California, Irvine, School of Information and Computer Science, 2010. [Online]. Available: <http://archive.ics.uci.edu/ml>.
- [58] S. A. Ong, H. H. Lin, Y. Z. Chen and Z. Cao, "Efficacy of Different Protein Descriptors in Predicting Protein Functional Families," *BMC Bioinformatics*, vol. 8, no. 8, pp. 267-280, 2007.
- [59] C. H. Q. Ding and I. Dubchak, "Multi-Class Protein Fold Recognition Using Support Vector Machines and Neural Networks," *Bioinformatics*, vol. 17, no. 4, pp. 349-358, 2001.
- [60] K. Kira and L. A. Rendell, "A Practical Approach to Feature Selection," in *Proceedings of the 9th International Workshop on Machine Learning*, Aberdeen, 1992.
- [61] I. Kononenko, "Estimating Attributes: Analysis and Extensions of RELIEF," in *Proceedings of the 1994 European Conference on Machine Learning*, Catania, 1994.

- [62] M. Robnik-Sikonja and I. Kononenko, "An Adaptation of Relief for Attribute Estimation in Regression," in *Proceedings of the 14th International Conference on Machine Learning*, Nashville, 1997.
- [63] H. Liu and R. Setiono, "Chi2: Feature Selection and Discretization of Numeric Attributes," in *Proceedings of 7th International Conference on Tools with Artificial Intelligence*, Herndon, 1995.
- [64] I. W. J. Guyon, S. Barnhill and V. Vapnik, "Gene Selection for Cancer Classification Using Support Vector Machines," *Machine Learning*, vol. 46, no. 1, pp. 389-422, 2002.
- [65] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization," in *Proceedings of the 15th International Conference on Machine Learning*, Madison, 1998.
- [66] S. I. Cessie and J. C. V. Houwelingen, "Ridge Estimators in Logistic Regression," *Applied Statistics*, vol. 41, no. 1, pp. 191-201, 1992.
- [67] C. Ratanamahatana, "CloNI: Clustering of Square Root of N-Interval Discretization," in *Proceedings of the 4th International Conference on Data Mining Including Building Application for CRM & Competitive Intelligence*, Rio De Janeiro, 2003.
- [68] Y. Yang and G. I. Webb, "Proportional k-Interval Discretization for Naive-Bayes Classifiers," in *Proceedings of the 12th European Conference on Machine Learning*, Freiburg, 2001.
- [69] W. Qiu and H. Joe, "ClusterGeneration: Random Cluster Generation (with Specified Degree of Separation)," *R Foundation for Statistical Computing*, vol. R Package Version 1.2.9, 2012.
- [70] J. R. Quinlan, "C4.5: Programs for Machine Learning," San Francisco: Morgan Kaufmann Publishers, 1993.
- [71] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, 1995.
- [72] C. Torrence and G. P. Compo, "A Practical Guide to Wavelet Analysis," *Bulletin of the American Meteorological Society*, vol. 79, no. 1, pp. 61-78, 1998.

- [73] R. Bracewell, "Rayleigh's Theorem: The Fourier Transform and Its Applications," Third Edition, New York: McGraw-Hill, 1999.
- [74] C. Legány, S. Juhász and A. Babos, "Cluster Validity Measurement Techniques," in *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, Madrid, 2006.
- [75] X. L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841-847, 1991.
- [76] M. Halkidi and M. Vazirgiannis, "Clustering Validity Assessment: Finding the Optimal Partitioning of a Data set," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, San Jose, 2001.
- [77] M. Halkidi, Y. Batistakis and M. Vazirgiannis, "On Clustering Validation Techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107-145, 2001.
- [78] Y. Shi, Y. Song and A. Zhang, "A Shrinking Based Clustering Approach for Multidimensional Data," *IEEE Transaction on Knowledge and Data Engineering*, vol. 17, no. 10, pp. 1389-1403, 2005.
- [79] I. Solomonovich Gradshteyn, I. Moiseevich Ryzhik, A. Jeffrey and D. Zwillinger, "Tables of Integrals, Series, and Products," Sixth Edition, San Diego: Academic Press, 2000.
- [80] S. Dua and S. Saini, "Data Shrinking Based Feature Ranking for Protein Classification," in *Information Systems, Technology and Management, series Communications in Computer and Information Science*, Heidelberg, Springer Berlin Heidelberg, pp. 54-63, 2009.
- [81] S. Saini and S. Dua, "A Grid-Based Scalable Classifier for High Dimensional Datasets," in *Information Systems, Technology and Management, series Communications in Computer and Information Science*, Heidelberg, Springer Berlin Heidelberg, pp. 404-415, 2010.