UNIVERSITY OF HERTFORDSHIRE

DOCTORAL THESIS

Using Feature Weighting as a Tool for Clustering Applications

Author:
Deepak PANDAY

Supervisor:
Dr. Peter LANE

Submitted to the University of Hertfordshire in partial fulfilment of the requirement of the degree of Doctor of Philosophy

October 13, 2021

Abstract

The weighted variant of *k*-Means (*Wk*-Means), which assigns values to features based on their relevance, is a well-known approach to address the shortcoming of *k*-Means with data containing noisy and irrelevant features. This research aims first to explore how feature weighting can be used for feature selection, second to investigate the performance of Minkowski weighted *k*-Means (*MWk*-Means), and its intelligent variant, on datasets defined in different p-norms, and third to address the problem of missing values with a weighted variant of *k*-Means. A partial distance approach is used to address the problem of missing values for weighted variant of *k*-Means.

Anomalous clustering has been successfully used to detect natural clusters and initialize centroids in *k*-means type algorithms. Similarly, extensive work has been carried out on using feature weights to rescale features under Minkowski L_p metrics for $p \geq 1$. In this thesis, aspects from both of these approaches enable feature weights to be detected based on natural clusters present in the training data, but the clusters are not limited to spherical shape. Two methods, *mean*-FSFW and *max*-FSFW, are developed as further extensions of intelligent Minkowski Weighted *k*-Means(*iMWk*-Means), where feature weights are used as indices for feature selection with no requirement for user-specified parameters.

The proposed feature selection methods are able to significantly reduce the number of noisy features. These methods are further extended to *mean*-FSFWext*PD* and *max*-FSFWext*PD* to address missing values and are found to be better alternatives than existing imputation methods.

The effect of feature weighting on clustering of dataset defined in varying p-norms is further explored in the thesis. An algorithm that translates a dataset into different p-norms has been proposed. The capability of *MWk*-Means to read true shapes of clusters defined in different p-norms is explored.

To address the problem of missing feature values in weighted variant of *k*-Means, different missing-value imputation methods are tested. The *MWk*-Means and its intelligent variant are further extended to incorporate the partial distance approach, specifically to address the problem of missing values.

All these methods are tested in both synthetic and real-world datasets against three models of noise - noisy feature added, feature blurring and cluster-wise feature blurring - where applicable. These noises are generated from Gaussian and uniform distribution with three different strength of noise, i.e., no noise, half noise and full noise

Overall, results demonstrate that feature weighting can improve feature selection. The partial-distance approach, with feature weights, is effective at ignoring missing values, and cluster retrieval in various p-norm spaces is effective.

Acknowledgements

First of all, I would like to express my sincere thanks to my supervisor Dr. Peter Lane for his consistent support, thoughtful comments and guidance with constructive criticism. I am also thankful to my initial supervisor Dr. Renato Cordeiro de Amorim, who helped me to grasp the correct way on the research.

I would also like to thank Dr. Na Helian for the 'continuous encouragement and willingness at all times to assist in any way throughout my research. I would also like to express my gratitude and appreciation to David Mayor for his insightful comments regarding my thesis writing and for the grammatical review of this thesis.

Last but not the least, this thesis would never be possible without the strong support of my family; in particular, a special thanks to my wife for her support both mentally and financially throughout the period.

Contents

Abstract **iii**

Acknowledgements **v**

Table of Contents **x**

List of Figures **xii**

List of Tables **xv**

Publication **xv**

Abbreviations **xvi**

1 Introduction **1**
1.1 Introduction and Objective . . . 1
1.2 Contribution . . . 2
1.3 Scope . . . 3
1.4 Thesis Structure . . . 3

2 Literature Review **5**
2.1 Topology of Clustering Algorithm . . . 5
2.2 Clustering Algorithms . . . 6
2.2.1 *k*-Means . . . 7
2.2.2 Initialization of Centroids . . . 7
2.3 Dimension Reduction . . . 8
2.3.1 Feature Selection . . . 8
Some Common Practice in Feature Selection . . . 9
Semi-Supervised Feature Selection . . . 10
2.4 Feature Weighting . . . 10
2.5 Missing Values . . . 12
2.6 Lp Norm . . . 14

3 Tools, Techniques and Methodologies **15**
3.1 Similarity Measurement . . . 15
3.2 p-Norms and L_p space . . . 16
3.3 Data Standardization . . . 16
3.4 Feature Weighting . . . 17
3.5 *k*-Means Algorithms and its Weighting Variant . . . 19
3.5.1 Generic *k*-Means . . . 19
3.5.2 *ik*-Means . . . 19
3.5.3 *Wk*-Means . . . 20
3.5.4 *MWk*-Means . . . 20
3.6 Cluster Validity . . . 21
3.7 Feature Selection . . . 22

3.7.1 intelligent K-Means Feature Selection(ikFS) . . . 22
3.7.2 Feature Selection using Feature Similarity . . . 23
3.7.3 Multi-Cluster Feature Selection . . . 24
3.8 Missing Data . . . 24
3.8.1 Missing Data Patterns . . . 24
3.8.2 Missing-data Mechanism . . . 26
3.8.3 Dealing with Missing Attribute Values . . . 27
Imputation Methods . . . 27
Attribute Mean Value . . . 27
k-Nearest Neighbour Techniques . . . 27
Regression Based Imputation . . . 30
No Imputation . . . 30
Partial Distance . . . 30
3.9 Dataset . . . 31
3.9.1 Synthetic Gaussian Distribution . . . 32
3.9.2 Real-world Dataset . . . 32
3.9.3 Noise . . . 34
Noise Strength . . . 34
Noise Distribution . . . 34
Noise Model . . . 35
3.9.4 Noise Configuration . . . 35
3.9.5 Datasets and Noise Configurations for Experiments . . . 37

4 Proposed Algorithms **39**
4.1 Feature selection . . . 39
4.1.1 Feature selection via Feature Weighting . . . 39
4.2 Partial Distance for Handling Missing Values . . . 39
4.2.1 wk-MeansPD . . . 39
4.2.2 Minkowski Weighted *k*-Means with partial distance(*MWk*-Means*PD*) . . . 40
4.3 Transformation of Dataset to Different p-norms . . . 40
4.3.1 Transformation of a Dataset from p_{old}-norm to p_{new}-norm . . . 41

5 Feature Selection **43**
5.1 Aims . . . 43
5.2 Experimental Set-up . . . 44
5.2.1 Dataset Set-up . . . 44
5.2.2 Parameter Tuning . . . 44
5.3 Result Presentation . . . 45
5.3.1 Organization of Results . . . 45
5.3.2 Results from Synthetic Datasets . . . 45
5.3.3 Organizing from Real-World Datasets . . . 45
5.3.4 Table Organization . . . 46
5.3.5 Synthetic Datasets . . . 47
5.3.6 UCI datasets . . . 53
5.4 Conclusion . . . 59

6 Prediction of Missing Values **61**
6.1 Aims . . . 61
6.2 Experimental Set-up . . . 62
6.2.1 Setting up Dataset . . . 62
6.2.2 Addition of Missing Values . . . 62
6.2.3 Selection of Algorithms . . . 63

6.2.4 Extension to Noise . . . 63
6.3 Reading Results . . . 63
6.3.1 Organization of Results . . . 64
Results for Imputation Methods . . . 64
Results from Different Approaches . . . 64
6.4 Experimental Results . . . 64
6.4.1 Imputation of Different Algorithms for Weighted *k*-Means and Intelligent Weighted *k*-Means . . . 65
Missing values with MCAR Mechanism . . . 65
Missing values with NMAR Mechanism . . . 71
6.4.2 Evaluation of Partial Distance against Different Methods in *k*-Means and Weighted *k*-Means . . . 77
Missing values with MCAR Mechanism . . . 78
Missing values with NMAR Mechanism . . . 85
6.5 Conclusion . . . 92

7 Performance of *k*-Means types in p-norms **95**
7.1 Aims . . . 95
7.2 Validation of Transformation Process . . . 96
Visualization of Transformation Process in 2D . . . 98
7.2.1 Generation of a Gaussian Mixed Model in Different p-norms . . . 103
7.3 Effect of p-norms over Clustering Algorithms . . . 106
7.3.1 Performance of Different Aggregation Functions . . . 106
Average Cluster Recovery . . . 107
p_{dist} equals to p_{plane} . . . 108
p_{dist} equals to 2 . . . 109
p_{dist} that yield max ARI . . . 110
7.3.2 Comparison of Different Aggregation Functions . . . 111
7.3.3 p-norms vs Distance Coefficient . . . 113
p_{dist} vs p_{plane} in *MWk*-Means . . . 114
p_{plane} vs p_{dist} in *iMWk*-Means . . . 114
7.4 Conclusion . . . 115

8 Extended Experiments **117**
8.1 Experimental set-up . . . 117
8.2 Extension of feature selection . . . 118
8.2.1 Percentage of noisy feature selection . . . 118
Synthetic datasets . . . 118
Real-world (UCI) datasets . . . 118
8.2.2 Extending the method to larger datasets . . . 119
8.2.3 Performance of classifiers after feature selection . . . 120
8.2.4 Performance of feature selection in L_p space . . . 123
8.3 Extension of partial distance to the Minkowski metric . . . 125
8.4 Conclusion . . . 125

9 Discussion and Conclusion **127**
9.1 Research outcomes . . . 127
9.2 Observations . . . 128
9.3 Limitations . . . 128
9.4 Further research . . . 129

A Coordinates System **141**
A.1 Cartesian to polar conversion . . . 141

B Gaussian Distribution **145**
B.0.1 Standard Gaussian Distribution . . . 145
B.0.2 Gaussian Distribution in two Dimension . . . 146
B.0.3 Generalization of Gaussian Distribution . . . 146

C Gaussian Mixed Model **149**
C.0.1 Gaussian Mixed Model in one Dimension . . . 149
C.0.2 Gaussian Mixed Model (GMM) for two Dimensions . . . 150

D GMM Control Parameters and k **153**
D.1 Control Parameters in GMM . . . 153
D.1.1 Visualization of GMM Datasets . . . 153
D.1.2 Observation based on cluster recovery . . . 156

E Effect of Noise in k-means **159**
E.1 Using Cluster Validation Index in L_p . . . 160

List of Figures

2.1 Topology of clustering algorithms 5
2.2 Taxonomy of dimension reduction [1]. 8

3.1 Different shapes of clusters, in two dimensions, defined by distance coefficient p, are displayed in 2-norm. Each cluster has a spherical shape in the particular p-norm i.e. data points lying on the circumference of the cluster are equidistant from its centroid. This figure is taken from the public domain https://en.wikipedia.org/wiki/Minkowski_dis
3.2 A cluster of data points which have higher dispersion along x-axis than y-axis is created using the GUI where + and * represent data points and centroids respectively. 18
3.3 Two different clusters- red and green are created using the GUI. Red cluster has data points with higher dispersion along x-axis whereas green cluster has data points with higher dispersion along y-axis. 18
3.4 Examples of missing-data patterns. Rows and columns correspond to observations and variables respectively. Data rows and columns are sorted according to the pattern of missing data. 25
3.5 Examples of missing data mechanism. 26
3.6 An example set for complete case kNN (*CC*-kNN) imputation in a dataset with univariate missing pattern and MCAR missing mechanism to impute missing values in entity y_1 28
3.7 Example of incomplete case kNN, *iC*-kNN in a dataset with univariate missing pattern with MCAR missing mechanism. 29
3.8 Uniform noise generated in the range [0,1]. 34
3.9 Noise generated from Gaussian distribution with mean = 0 and standard deviation = 1. 34
3.10 Examples of adding noise in dataset. 35
3.11 Noise configuration: there are two noise distributions, two noise strength and three noise models. 36

7.1 Transformation of a 2D cluster with center at (10, 5) from 2-norm space to 0.5-norm space. The process is carried out in six steps from top left (a) to bottom right (e). 97
7.2 Results from moving data points, originally defined in 2-norm with center at 10,5), to p-norms ($p \in [0.05, 1, 1.5]$ are displayed in each of the sub-plots on left hand side. Results from the reverse process are displayed sub-plots on the right. 99
7.3 Data points with center (10,5) defined in two dimensions in Euclidean space is reshaped using a different distance coefficient (p). Each sub plot in the left column represents the data points changed to its respective p-norms. The right column shows the results from reversing the process back to a 2-norm. 101
7.4 A set of data points with center (10,5) defined in the two dimensions in Euclidean space is reshaped using some extreme distance coefficient (p). The values of p are either less than 0.005 or greater than 100. To the right of each sub-plot with a given p, a reverse sub plot shows the results obtained when the process is reversed. 103

7.5 First and second PCA components view of three Gaussian clusters generated using the MATLAB Gaussian mixed model. 104
7.6 First and Second PCA component view of 3 Gaussian clusters generated using the MATLAB Gaussian mixed model. Centroids for each of these clusters are generated from a Uniform distribution and the GMM has a spherical shape. 105
7.7 Performance of four different *k*-Means Type algorithms on GMM datasets defined in different p-norms, p_{plane} ranging from 0.5 to 5. 108
7.8 Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where distance coefficient p_{dist} is equal to p-norm, p_{plane}. 109
7.9 Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where the distance coefficient p_{dist} is equal to 2. 110
7.10 Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where the distance coefficient, p_{dist} is considered the one which maximize the ARI. 111
7.11 Performance of four different combinations of the distance coefficient, p_{dist} used in *MWk*-Means (p1) over different datasets defined using a range of p-norms. . . . 112
7.12 Performance of four different combination of the distance coefficient, p_{dist} used in *IMWk*-Means for different datasets defined for a range of p-norms, p_{plane}. 113

A.1 Polar coordinate system in two dimensions. 141
A.2 Polar coordinate system in three dimensions. 142

B.1 Probability density function (PDF) of Gaussian Model with mean = 0 and sigma (standard deviation) = 1. 145
B.2 Probability density functions (PDFs) of two Gaussians with two variate random variables x and y. 146
B.3 Normalized Gaussian distribution in three dimension. 147

C.1 Probability density functions (PDFs) of three univariate Gaussian components . . 149
C.2 Mixed model Gaussian (GMM) distribution of three univariate Gaussian components is given by a linear combination of the three Gaussians. 150
C.3 Probability density functions (PDF) of two variate Gaussian components 150

D.1 Visualization of the synthetic datasets generated by Gaussian mixed models. These models consist of two GM components and have four variates (features). First two PCA)Principal component analysis) component are used to display the data points.154
D.2 Visualization of the synthetic datasets generated by spherical shaped Gaussian mixed models. These models consist of four Gaussian components and have four variates (features). First two components of PCA are used to display data points. . 155
D.3 Performance of *k*-Means is influenced by the number of features of the Gaussian components. 156
D.4 Performance of *k*-Means against the number of Gaussian components. 157
D.5 Performance of *k*-Means against the covariance of the Gaussian components. . . . 158

E.1 Effect of noise on the performance of *k*-Means. Datasets are generated using Gaussian mixed models. Noise i generated from two different distributions: Gaussian and uniform distributions. 159
E.2 Effect of noise on the performance of *k*-Means in real-world datasets. Datasets are generated using Gaussian mixed models. Noise are generated from two different distributions: Gaussian and uniform distributions. 159

List of Tables

3.1 The experiment of synthetic mixed-model Gaussian distribution contains 20 different datasets with 1000 data instance in each. The distribution has spherical Gaussian clusters with diagonal covariance matrices of 0.5 32
3.2 Real-world datasets used in our experiments 33
3.3 Australian Credit Dataset 33

5.1 Comparision of four different feature selection algorithms with original var of *k*-Means and intelligent Minkowski weighted *k*-Means in synthetic datasets with and without added noise. Silhoutee index is used for parameter tuning. 47
5.2 Comparison of four different feature selection algorithms with original variate of KMeans and intelligent Minkowski Weighted KMeans in synthetic dataset under the supervised condition (best possible cases). 50
5.3 Comparision of three values of distance coefficient(p) in the purposed feature selection algorithms. The performance is evaluated using ARI index and are carried out in Gaussian mixed-model. 52
5.4 Comparision of four different feature selection algorithms with original variate of kMeans and intelligent Minkowski Weighted KMeans in real-world dataset with and without added noise where Silhoutee index is used for parameter tuning. . . . 54
5.5 Comparison of four different feature selection algorithms with original variate of KMeans and intelligent Minkowski Weighted KMeans in real-world dataset under the supervised condition (best possible cases). 56
5.6 Comparision of the purposed feature selection algoirthm with three different values of distance coefficient(p). The performance is evaluated using ARI index and are carried out in Gaussian mixed-model. 58

6.1 Comparison of four different imputation methods in Gaussian mixed model where missing values have MCAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset. 65
6.2 Comparison of four different imputation methods in Gaussian mixed models where missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has highest correlation with its dataset. 67
6.3 Comparison of four different imputation methods in the real-world datasets where missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset. 68
6.4 Comparison of four different imputation methods in the real-world (UCI) datasets where missing values have MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 70
6.5 Comparison of four different imputation methods in Gaussian mixed models where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has the lowest correlation with its dataset. 72
6.6 Comparison of four different imputation methods in Gaussian mixed models where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 73

6.7 Comparison of four different imputation methods in the real-world datasets where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset. 75
6.8 Comparison of four different imputation methods in the real-world datasets where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 76
6.9 Comparison of two difference clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset. 78
6.10 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 80
6.11 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset. 82
6.12 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 84
6.13 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset. 86
6.14 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in a Gaussian mixed model. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset. 87
6.15 Comparison of two difference clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset. 89
6.16 Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset. 91

7.1 Performance *MWk*-Means in GMM datasets defined in a range of p-norms. 114
7.2 Performance *iMWk*-Means in GMM datasets defined in a range of p-norms. 115

8.1 The average percentage of noisy features selected, and their respective standard deviations, in synthetic datasets 118
8.2 The average percentage of noisy features selected, in the real-world (UCI) datasets. 119
8.3 Performace of feature selection algorithms in UCI datasets with a higher degree of feature space cardinality. 120
8.4 The average accuracy of three classifiers, and respective standard deviations, on synthetic data sets containing about 50% or 100% extra features composed of uniformly random noise. 121

8.5 The average accuracy of three classifiers, and their respective standard deviations, in real-world data sets containing about 50% or 100% extra features composed of uniformly random noise. 122
8.6 The average adjusted Rand Index and its standard deviation for the cluster recovery in synthetic datasets with and without noisy features in different p-norms. Silhouette Index is used for parameter tunning. 124
8.7 The average adjusted Rand Index and its standard deviation for the cluster recovery in synthetic datasets defined in p-norms. The datasets have missing values in first two features generated using the missing completely at random (MCAR) mechanism. The Performance columns contains mean/std of the ARIs. 125

E.1 Performance of kMeans variants in different L_p. The Performance column contains mean/std 161

Publications

Panday, D., de Amorim, R.C. and Lane, P., 2018. Feature weighting as a tool for unsupervised feature selection. Information processing letters, 129, pp.44-52.

D. Panday, P.C.R. Lane and N. Helian, Using feature weighting as a tool for clustering applications, in L. Coudron (Ed.), Proceedings of Abstracts Engineering and Computer Science Research Conference, pp.47-49, 2019.

List of Abbreviations

AP	**A**nomalous **P**attern
ARI	**A**djusted **R**and **I**ndex
BIC	**B**ayesian **I**nformation **C**riterion
BMDP	**B**io-**M**edical **D**ata **P**ackage
DAG	**D**irected **A**cyclic **G**raph
DSRMR	**D**ual **S**elf **R**epresentation **M**anifold **R**egularization
GMC	**G**aussian **M**ixed **C**luster
GMM	**G**aussian **M**ixed **M**odel
GRNSR	**G**raph **R**egularized non-**N**egative **S**elf **R**epresentation
GUI	**G**raphical **U**ser **I**nterface
HINoV	**H**euristic **I**dentification of **N**oisy **V**ariables
FSFS	**F**eature **S**election using **F**eature **S**imilarity
FSFW	**F**eature **S**election using **F**eature **W**eighting
ISO-DATA	**I**terative **S**elf-**O**rganizing **D**ata **A**nalysis **T**echniques
MCA	**M**issing **C**ompletely at **R**andom
MCFS	**M**ulti-**C**luster **F**eature **S**election
ME	**M**aximization **E**xpectation
NMAR	**N**ot **M**issing **C**ompletely at **R**andom
PCA	**P**rincipal **C**omponent **A**nalysis
PD	**P**artial **D**istance
PDF	**P**robability **D**istribution **F**unction
LS	**L**east **S**quare
LLS	**L**ocal **L**east **S**quare
RC	**R**egression **C**luster
SAS	**S**tatistical **A**nalysis **S**ystem
SPSS	**S**tatistical **P**ackage for the **S**ocial **S**cience
SRFS	**S**emi-supervised **R**epresentative **F**eature **S**election
SVD	**S**ingular **V**alue **D**ecomposition
SYNCLUS	**SYN**thesized **CLUS**tering

Chapter 1

Introduction

1.1 Introduction and Objective

The amount of data stored and processed by modern computers is increasing exponentially with advances in technologies and storage capacity. These huge amounts of data, also known as big data, grow not only in the number of instances, but also feature space, as more information about each data instance can be captured than before. Although this growth in feature space increases the amount of information known about the data, not all acquired features will have the same contribution to the information required when trying to understand the data. for example, information about the colour of a vehicle will not affect our understanding of its speed of motion.

A variant of *k*-Means, weighted *k*-Means(*Wk*-Means), is one of the approaches where bias is added to features in the form of feature weighting to minimize the influence of noisy and irrelevant features. Basically, the weighted variant of *k*-Means bases feature weights on the dispersion of features within clusters. In addition to the feature weighting approach, the Minkowski variant of weighted *k*-Means, *MWk*-Means, uses Minkowski metric for similarity measurement and is able to retrieve clusters other than those of a spherical shape. One of the research areas considered here is to take advantage of these two approaches and derive automated feature selection procedures so that clustering structures are preserved, and noisy and irrelevant features are removed after feature selection [2].

Missing values, which are inherent in the data collection process, are a common problem in data pre-processing. The two most common practices to address missing values in data mining are deletion of instances containing missing values and imputation of missing values, usually with the corresponding feature's average value [3]. Deletion of instances containing missing values results in loss of information, whereas feature average imputation decreases its feature's diversity. The missing value problem can be addressed implicitly by modifying a data processing algorithm so that it can handle missing values. In clustering algorithms, modifying the distance metric by its partial variant has been attempted to address the missing value problem. In this thesis, we examine the impact of using a partial distance approach in the weighted variant of *k*-Means. Similarly, *MWk*-Means is modified to incorporate partial distance within its similarity measurement framework.

Based on the three major areas of the research highlighted above, this thesis addresses the following questions

(I) Feature weighting as an index for feature selection

   a) Can feature weighting be used as a tool for feature selection?

   b) Can this approach be used to identify and remove different types of noisy features?

(II) Effect of missing values in the weighted variant of *k*-Means

a) How do different imputation methods affect the performance of weighted variant of *k*-Means?
b) Can the partial distance approach be used in the weighted variant of *k*-Means to address missing values?
c) Can the partial distance approach be effectively applied to feature selection techniques based on feature weighting?

(III) Effect of clustering in different p-norms

a) Is it possible to translate a dataset defined in one *p*-norm to another *p*-norm and reverse the process?
b) Is feature weighting still effective to find clustering in the dataset, defined in different p-norms?
c) Is Minkowski metric still effective to find clustering in the dataset, defined in different p-norms?
d) Can feature selection based on feature weighting be effective with datasets in different p-norms?

1.2 Contribution

Feature weighting and Minkowski metrics are used in this thesis to explore three areas of clustering analysis: feature selection, addressing missing values and data in different p-norms using a weighted variant of *k*-Means. These areas of research are explained in Chapters 5, 6, and 7 respectively. Chapter 8 consists of results from the extended experiments which combine two or more of these areas. In general, the contributions are:

- Feature selection
  - Reviewed different feature selection algorithms.
  - Evaluated the proposed algorithms based on cluster recovery, the number of original features selected and the number of noisy features selected.
  - Demonstrated that the proposed feature selection algorithms performed better than other feature selection algorithms.
  - Highlighted how the distance coefficient(p) contributed to the performance of the proposed feature selection algorithms.
  - Evaluated the performance of the proposed feature selection algorithms in different classifiers.
- Missing values.
  - Reviewed different approaches used to handle missing values in clustering analysis.
  - Explored the effectiveness of different imputation methods for the weighted variant of *k*-Means.
  - Modified *Wk*-Means, *iWk*-Means and *MWk*-Means to incorporate partial distance metric within them to address missing values.
  - Demonstrated the impact of using partial distance in the weighted variant of *k*-Means.
- Working with data in different p-norms
  - Defined a translation process to move data from one p_{old}-norm to a second p_{new}-norm.
  - Explored the effectiveness of Minkowski metric and feature weighting for cluster analysis of data in different p-norms.
  - Explored the performance of the proposed feature selection algorithms in L_p space.

1.3 Scope

Three major scopes of this research are:

- Feature selection based on feature weighting addresses problems typical in big-data problems where instances are represented with large numbers of dimensions (known as the "curse of dimensionality").
- The partial distance approach provides a better alternative to imputation methods in *k*-Means type algorithms. This enables different data analysis tools to avoid the imputation of false values.
- The procedure to create the Gaussian mixed models in L_p space provides a new tool, enabling the data scientist to create a range of synthetic datasets.

1.4 Thesis Structure

Chapters in this thesis are structured as literature review, methodologies, and datasets are covered in Chapters 2 and 3 respectively; all purposed algorithms and methodologies are presented in Chapter 4; Chapters 5, 6, and 7 cover the three areas of research as mentioned in contribution; Chapter 8 consists of extended research which combines these three areas of research; and finally, the thesis is concluded with outcomes and discussions for further expansion in Chapter 9.

Chapter 2 covers a literature review on different approaches used in clustering analysis which includes the development of k-Means type algorithms and different feature selection algorithms. In this Chapter, we observed different approaches which are used to address feature weighting, find the number of clusters and initialize centroids in k-Means type algorithms. Literature on different feature selection algorithms based on cluster analysis is also studied.

In Chapter 3, different tools and methods used in this thesis are introduced and their mathematical models are explained. This chapter covers different metrics for similarity measurement, data standardisation, the optimization problem in different variants of *k*-Means, feature weighting principle and its use in cluster analysis, cluster validation indices, feature selection algorithms, pattern and mechanism of missing values, different imputation methods and partial distance approach for missing values; and p-norm. A short description of synthetic and real-world dataset, and different types of noise used in the experiments are presented in this chapter. This chapter also includes all proposed algorithms and novel approaches conducted in this thesis.

The Chapter 4 introduces all proposed methods, whereas, in Chapter 5, consists the experimental results from the proposed feature selection algorithms . The performance of these feature selection algorithms is evaluated based on cluster recovery and the number of original features retrieved with noisy features against three other feature selection algorithms.

The performance of different imputation methods to address missing values in *k*-Means type algorithms is observed in Chapter 6. This chapter also introduces the partial distance approach in the *k*-Means type algorithms.

Chapter 7 introduces different parameters of clustering analysis and explores the performance of the *k*-Means type algorithm in p-norms. This chapter explains the proposed methods for transforming data from p_1-norm to p_2-norm and analyses the performance of Minkowski weighted *k*-Means, *MWk*-Means in different p-norms.

Chapter 8 combines the three areas of research where results from extended experiments including the performance of the purposed feature selection algorithms in a large dataset, their effect on different classifiers, and L_p space are presented and analysed.

Chapter 9 concludes the contributions of this thesis. Limitations of the proposed algorithms and further research areas are discussed in this chapter.

Chapter 2

Literature Review

Clustering is a basic tool in data-preprocessing where “like" items are grouped together and “unlike" are separated from each other. The clusters help disclose fundamental structures in the data such as the characteristic of objects and their relationships. Clustering is a form of unsupervised learning where information is extracted based on the hidden structure of data. This is in contrast to supervised learning which aims to create a classifier model with the help of class labels. Clustering has been used in different data processing applications such as data mining, bioinformatics [4, 5, 6], image processing, numerical taxonomy [7], factor analysis [8], pattern recognition [9], web mining [10, 11, 12, 13, 14, 15] etc. Some of the earliest work in clustering includes grouping for quantitative analysis of the tribal cultures by Driver and Krobber [16] where groups of people are formed based on similarity. In their work, people are ranked “higher" or “lower" based on similarities they possess for tribes, and classes are derived using these ranked values.

2.1 Topology of Clustering Algorithm

Sneath and Sokla [7] have broadly divided clustering algorithms into two categories: hierarchical and non-hierarchical. The major distinction between hierarchical and non-hierarchical clustering lies in their methods and structures. In hierarchical methods, clusters form a nested tree (dendrogram) where a parent-child structure preserves the relationship between the clusters. A dendrogram helps to visualize the building process of the clusters and can be used to obtain any required number of clusters. Hierarchical clustering is more often used to study taxonomies in biological, and social and behavioural science [15].

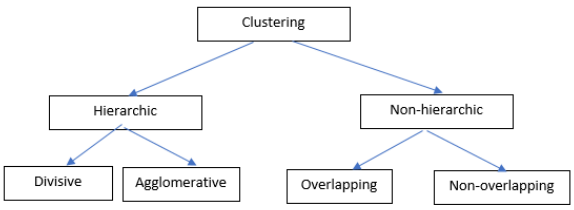


FIGURE 2.1: Topology of clustering algorithms

Hierarchical clustering can be further divided into agglomerative and divisive, based on the way they are created. In the case of agglomerative, each element is considered as a singleton cluster at bottom level (leaves) and a nested tree is formed by repetitive merging of the clusters until the complete dataset is recovered at the root. In divisive methods, the direction of cluster building is reversed, starting at the root with the complete dataset and carrying out repetitive divisions of clusters to children clusters until singleton clusters (leaves) are formed at the bottom end.

The non-hierarchical, also called partitional, clustering techniques are more common in engineering where a dataset is divided into multiple partitions which can be either overlapping or non-overlapping. In the case of overlapping partitions, a dataset is partitioned into clusters such that a single object can belong to multiple clusters with a degree of belonging, m_{ik}, also called a membership function, whereas in non-overlapping partitions, an object belongs to only one partition at a time. Overlapping clustering can be viewed as a generalization of non-overlapping clustering where membership function m_{ik} can either be 0 or 1.

Different partition based clustering algorithms have been put forward in the literature [17, 18]. For example, [17] has listed four methods: *k*-Means variant, Exchange, Seriation, and Graph Partitioning. *k*-Means, also called the moving centres, was originally developed by MacQueen [19] and a variant of *k*-Means, ISO-DATA (**I**terative **S**elf-**O**rganizing **D**ata **A**nalysis **T**echniques) was put forward by Ball and Hall [20]. In addition to the basic *k*-Means algorithm, ISO-DATA breaks down the largest cluster or merges too close clusters once the moving centres, i.e. centroids, are saturated. Moving centres and agglomerative type clustering algorithms have been used in different statistical packages, including BMDP, SPSS and SAS.

2.2 Clustering Algorithms

A clustering algorithm aims to partition a dataset Y, composed of n entities $y_i \in R^V$, into K homogeneous clusters where the feature space $V = \{v_1, v_2, ..., v_M\}$ consists of M features. Partition based clustering produces K disjoint clusters $S = \{S_1, S_2, ..., S_K\}$ so that $S_i \cap S_j = \emptyset$ for $i \neq j$. Fuzzy *c*-Mean, also called soft clustering is a generalization of *k*-Means. Unlike k-means where an element belongs to a single cluster a time, in fuzzy c-Means an elements partially belongs to all clusters as defined by the membership function m_{ik} such that $\sum_{k=1}^{K} = 1$.

A hierarchical clustering algorithm (e.g. Wards [21]) generates a dendrogram of clusters as well as additional information regarding the relationship between clusters. In hierarchical clustering, an entity y_i can be assigned to more than one cluster as long as they are at different levels in the dendrogram. Through the dendrogram, relationships between the clusters at each level can be derived. DENsity based CLUstEring (DENCLUE) [22], a density based clustering algorithm, is a better alternative to partitional based clustering which works on segmentation-based object categorization.

This research will focus on three main areas of clustering- feature selection, handling missing values, and analysis of clustering in a dataset defined in different p-norms other than 2-norms. Two most popular feature selection algorithms from Mitra et al. [23] and Deng Cai et al. [24] are based on similarity index and required parameter tuning which is not possible under totally unsupervised conditions. Since these are based on similarity index and try to peak most unique features, the addition of noisy features or blurring features with making them unique is likely to be selected. The cluster-based feature selection method put forward by Renato et al. [25], used feature weighting to cluster feature (instead of entities) but the selection of features from the cluster of features so formed are based on Mitra feature similarity index. Therefore, it again inherited the problem of uniqueness with noisy features. This thesis aims to overcome the problem by using only feature weighing in the sole for identifying the true features.

k-Means and its variants provide a basis of clustering algorithms and are easy to understand. In addition, there are a large number of research articles on *k*-Means and its variants in the area of concern. Moreover, due to the primitive nature of k-Means type algorithms, the finding can be easily practiced in other types of clustering. Therefore, this research will initially focus on the variants of *k*-Means.

2.2.1 *k*-Means

According to Cormack, clusters must maintain homogeneity within themselves and heterogeneity between clusters [26]. To satisfy the definition different authors tried to minimize within-cluster variation [27, 28, 29]. *k*-Means, one of the initial algorithms which fit in this category, was independently developed by Sebestyen [20] and MacQueen [19]. Because of its simplicity, easy of implementation and speed, *k*-Means soon became popular and has been used in different areas of data mining, such as cluster analysis [30, 27], multivariate analysis [31], pattern recognition [9], etc.

k-Means does have some shortcoming such as: (i) the value of K, number of clusters, has to be known beforehand; (ii) it may get trapped in local minima; (iii) it is biased towards spherical clusters; (iv) it provides no resistance against noisy or irrelevant features. This thesis is particularly focused on weaknesses (ii), (iii), and(iv).

In generic *k*-Means, initially K centroids are chosen at random and dataset is recursively partitioned to minimize within-cluster variation. During each iteration, centroids are reallocated to the center of the gravity of the corresponding clusters. Therefore, the performance of the *k*-Means depends on initial location of centroids, leading to the shortcoming (ii). Similarly, *k*-Means uses Euclidean distance as a similarity measurement which inherits bias towards spherical clusters. *k*-Means treats each feature equally, however, different features may have different degrees of relevance, for example a noisy or redundant feature has no contribution towards clustering and therefore can be considered as a feature with zero degree of relevance.

2.2.2 Initialization of Centroids

The performance of *k*-Means is highly influenced by the location of the initial centroids. Therefore, the algorithm is not always guaranteed to find the global optimum [32, 19] and its convergence depends on how well the initial centroids are chosen [33, 34]. One of the alternatives to avoid trapping in local maxima is to repetitively run *k*-Means with different centroids and accept the best solution [35, 27]. This can be practically impossible to find global optimum [6] since there are always chance to have a large number of local optima.

In some software packages [36], datasets are initially partitioned into K disjoint sets and initial centroids are defined as the mean of these partitions. *Pk*-Means [37], *k*-Means with a partition based cluster initialization method, first divides each feature into K equi-sized partitions and then one of the partitions, which is not selected earlier, is chosen randomly from each feature. A random value is drawn from the selected partition for each feature which gives one of the initial centroids. This process is repeated for remaining K-1 initial centroids.

Some deterministic methods for initialization of centroids have been put forward by different authors. For example, Astrahan [38] proposes a deterministic method which first finds the number of neighbouring points falling within the radius of a pre-defined distance d_1 for all entities. The entity with the highest number of neighbours is chosen as the first initial centroid. The next entity with maximum number of neighbours and at least d_2 distance from the first initial centroid is chosen as the second initial centroid. And the process is repeated until K initial centroids are derived.

Many researchers [39, 40] have followed Milligan's [41] approach of using hierarchical clustering, like Ward's method [42], for initialization of the centroids. Constrained *k*-Means [43], a semi-supervised *k*-Means, uses the labelled examples to draw initial centroids for the *k*-Means.

k-Means++ [44], a variant of *k*-Means, uses a probability based "D^2 weighting" method for initialization of centroids in *k*-Means. The convergence of *k*-Means++ is much faster than classical *k*-Means [44] and is popular in different literatures [45, 46, 47].

2.3 Dimension Reduction

Dimension reduction is a common pre-processing step in data analysis. There are different reasons for this, including: (i) helps to save processing time; (ii) reduces the disk space required to process and store data; (iii) allows to create more meaningful visualisation aids; and (iv) reduces issues raised by the *curse of dimensionality* [48, 49, 50, 51]. More importantly, it helps to get rid of redundant or irrelevant noisy features which may be misleading in further decision processing.

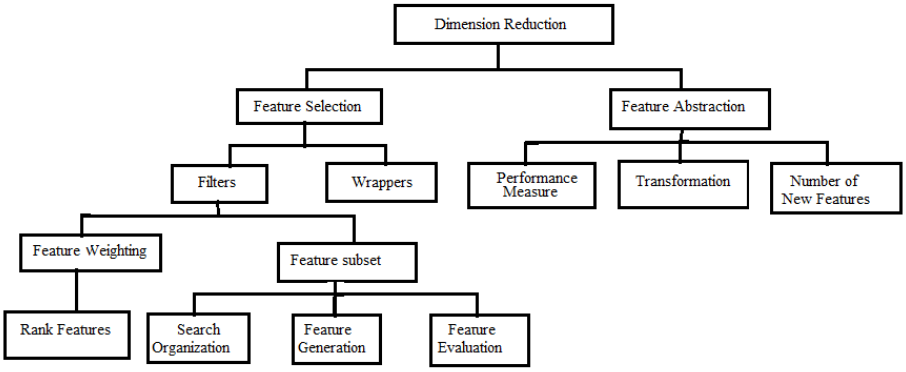


FIGURE 2.2: Taxonomy of dimension reduction [1].

Generally speaking, there are two main classes of methods for dimension reduction: feature selection and feature extraction as shown in Figure 2.2. Feature selection can be further classified as filters and wrappers methods whereas feature abstraction can be carried out based on performance measures, transformation or number of new features. Feature selection attempts to find the smallest subset of relevant features, according to a given criterion. Feature selection methods do not alter the features themselves, preserving their original meaning to the user. Feature extraction attempts to reduce the dimensionality of data sets by combining features. The original features and their meaning to the user are usually lost during this process although feature extraction aims to minimise the loss of information.

2.3.1 Feature Selection

The main objective of feature selection algorithms is to determine a proper subset $V' \subset V$, so that describing an entity $y_i \in Y$ over feature subspace V' instead of V does not lead to loss of information. Feature selection has a long history (see for instance [52, 53, 54, 55, 56] and references therein). However, this tends to be for algorithms following the supervised learning framework which requires labelled data.

Feature selection algorithms basically consist of two major components: feature search - finding the subset of features and feature evaluation - evaluation of feature/feature subset based on predefined criterion [57]. Sequential search like sequential forward/backward selection, bidirectional selection [58, 59] have been practised in different literatures. Most of these sequential search are based on greedy techniques and hence cannot guarantee an optimal solution [57]. To avoid becoming trapped in local maxima, some authors have used random searching methods like genetic algorithms and random mutation hill climbing [60, 61]. Once the feature search is completed, irrelevant or redundant features are removed and "informative" features are selected

from the feature search space based on some feature criterion functions. In filter approach, feature evaluation is purely based on the data itself whereas in case of wrapper, selection of the features is based on how informative the particular feature is for the chosen clustering algorithm. Addition of extra steps makes the wrapper approach comparatively slower than filter approach but maximizes the information. However, it is limited to the inbuilt cluster evaluation algorithm.

Some Common Practice in Feature Selection

A forward selection technique, proposed by Fowlkes et al. [62], is able to find a subset of “meaningful" features for complete linkage hierarchical clustering but can be extended to *k*-Means type partition clustering methods as well. In cluster analysis of market segment, Carmone et al. [63] explored how the selection of the “best" subset is important, as “noisy" features can be misleading. The proposed feature selection algorithm, the Heuristic Identification of Noisy Variables(HINoV) [63], is based on the assumption that the adjusted rand index (RAND) is larger for “true" features and smaller for “masking" features. HINoV is found to be effective for *k*-Means cluster analysis of data-based market segments. Brusco et al. [63] discovered that HINoV is not always effective: it is more informal and subjective as it requires interpretation of a screen plot. Moreover, HINoV assumption is not always valid since there can be high-degree of correlation between the masking variables and multiple sets of true cluster structure can persist in a single dataset. Brusco et al. address these problems with a new ARI-based variable selection heuristic, VS-KM, which builds on the basis of Carmone et al. [63] and Fowlkes et al. [62] for partitional based algorithms like *k*-Means. VS-KM works in a forward-selection manner by using between-cluster and sum-of-square information as suggested by Fowlkers et al [62]. VS-KM is tested with 2200 datasets with known configuration by Monte Carlo testing and it has been empirically demonstrated as effective in identifying “informative" variables.

Dash et al. [64] used entropy base feature ranking method to remove irrelevant and redundant features from 17 different real-world datasets containing both continuous and nominal data types. In their experiment, they have used a sequential backward selection algorithm for searching “informative" features and shown that the performance of their method is almost close to the supervised feature ranking method, RELIEF [65].

Vandenbroucke et al. [66] used a competitive learning algorithm based on colour texture feature to find the most “informative" colour texture feature for soccer image segmentation. The competitive learning scheme first derives candidate features based on their contribution to the junction function derived from the intra-cluster compactness and between-cluster dispersion. The most “informative" features are then selected from the candidate features using the correlation coefficient between the junction function and the candidate features. The performance of this approach is found to be much closer to the supervised method considered.

For large datasets, both in size and feature space, Mitra et al. [23] developed an unsupervised feature selection algorithm based on feature similarity measurement called maximum information compression index. The algorithm recursively divides the feature space into ‘K’ groups under KNN principle and the feature with the most compact neighbours are selected based on the compression index. The algorithm is much faster than its counterparts as it does not require feature search.

In recent years, spectral/spare learning has become popular in unsupervised feature selection. Multi-cluster feature selection, MCFS [24], is one of the earliest multivariate feature selection algorithm based on the spectral learning model. MCFS first detects the cluster structure using

spectral analysis [67] then the usefulness of each feature per cluster is derived using l_1-norm regularization [68]. The features with the highest coefficient are selected as the representative features.

Regularized self-representation, the RSR model is proposed by Zhu et al. [48] as a new spare learning model for unsupervised feature selection based on assumption that informative features can be presented or used as a linear combination of relevant features. RSR model used $l_{2,1}$-norm to characterize the representation matrix and the features with highest weight are selected as the representation feature. The extended version of RSR [69] uses $l_{2,p}$-norm to a derive representative matrix. Other related works on self-representation models are graph regularized non-negative self representation(GRNSR) [70], dual self-representation, manifold regularization(DSRMR) [71] etc.

Semi-Supervised Feature Selection

With the popularity of constraints/semi-supervised clustering [72, 73, 74, 75], different researchers have attempted to utilize partially available class labels or other constraints for feature selection. These techniques are mostly found in practice for filter methods [76, 77, 78] and most of these algorithms use some sort of score function like variance, Laplacian, Fisher, Constraint, etc. The semi-supervised filter feature, SRFS [77], is one of the frameworks that first derives relevant features by removing irrelevant and redundant features using mutual (entropy) information. Directed acyclic graph (DAG) is then constructed using the relevant features and then partition the graph into clusters based on the Markov blanket [79]. Finally, representative features from each cluster are derived.

2.4 Feature Weighting

Feature weighting is a generalization of feature selection [80]. In feature weighting, weight w_v, is assigned to the feature $v \in V$ based on its relevance. Feature selection can be modelled with feature weight by adding a constant that the weight value w_v is either 1 (selected) or 0 (not selected). In most algorithms,the weight values $w_v \in (0, 1)$ sum to 1, i.e. $\sum_{v \in V} w_v = 1$ whereas some algorithms may assign different feature weight values based on cluster w_{kv} such that $\sum_{v \in V} w_{kv} = 1$ where $k = 1, 2, ..., K$ is the number of clusters.
Sneath and Sokal in [7] discussed weighting factors in clustering and disclosed that some features are more significant than others. Fowlkers et al. [62] proposed an algorithm that considered the importance of features in clustering. The algorithm performs feature selection and feature subset evaluation in three steps: forward selection, backward elimination and guiding feature selection. Desarbo et al. [81] have proposed a new variant of *k*-Means, SYNCLUS, SYNthesized CLUStering which extends *k*-Means such that different degrees of relevance of features are considered while clustering. SYNCLUS, in addition to *k*-Means parameters (number of clusters and initial seeds i.e. centroids), requires two more parameters – information about the features grouping and initial feature weights in each group. The work has been extended to find optimal feature weighting for ultrametric and additive tree fitting [82, 83]. Desarbo and Mahajan [84] have extended this concept to constraints clustering where feature relevance is reflected with the linear transformation of the features.

Modha and Spangler [85] presented a framework that integrates multiple heterogeneous features in the *k*-Means. The extended variant of *k*-Means, convex *k*-Means clusters objects based on symmetric “distortion" between two objects along the feature vectors. The distortion between two entities is measured as

$$D_w(y_i, y_j) = \sum_{v \in V} D_v(y_{iv} - y_{jv}) \tag{2.1}$$

where the dispersion D_v depends on the feature space.

$$D_v(y_{iv}, y_{jv}) = \begin{cases} (y_{iv} - y_{jv})^T (y_{iv} - y_{jv}) & \text{for Euclidean feature sapce} \\ 2(1 - (y_{iv}^T y_{jv}) & \text{for Spherical feature space} \end{cases} \tag{2.2}$$

The convex *k*-Means replaces the similarity measurement of the classical *k*-Means by the weighted distortion and optimizes it using convex formulation. The feature weighting ("distortion") is altered in each iteration so that the average within-cluster dispersion is minimized and the average between-cluster dispersion is maximized. The objective function is to minimize the criterion:

$$F(S, C, w) = \sum_{k \in K} \sum_{v \in V} D_w(y_i - c_k) \tag{2.3}$$

Chang et al. [86] introduced an extension of *k*-Means, feature weighted *k*-Means(AWK), which considers that a feature can have different degrees of relevance in different clusters. AWK assigns a cluster-based feature weighting w_{vk} for a feature v ∈ V in cluster $S_k \in S$. AWK tries to optimize

$$F(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v \in V} w_{kv}^{\beta} d(y_{iv}, c_{kv}) \tag{2.4}$$

the weighting coefficient, $\beta > 1$ and the similarity metric is

$$d(y_{iv}, c_{kv}) = \begin{cases} |y_{iv} - c_{kv}|^2 & \text{for numerical data,} \\ 1 & \text{for categorical data with } y_{iv} = c_{kv}, \\ 0 & \text{for categorical data with } y_{iv} \neq c_{kv} \end{cases} \tag{2.5}$$

The feature weighting w_{vk} is initially set to $1/|V|$. The cost function F(S,C,w) is minimized by partially optimization of S, C and w.

A new variant of weighted *k*-Means, *Wk*-Means was proposed by Huang et al. [87] which is basically AWK discussed earlier except a single weight w_v is assigned to a feature $v \in V$ instead of cluster-based weights. Therefore, the objective function 2.4 is modified to

$$F(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v \in V} w_v^{\beta} d(y_{iv}, c_{kv}) \tag{2.6}$$

The objective function above is minimized by optimizing S, C, and w as in AWK and the feature weight w_v in each iteration which is calculated by

$$w_v = \begin{cases} \frac{1}{\sum_{u \in V} \frac{D_v}{D_u}^{(1-\beta)}} & \text{if } D_v \neq 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

where the dispersion of feature $v \in V$ is calculated as the sum of dispersion of the feature within all clusters,

$$D_v = \sum_{k=1}^{K} \sum_{i \in S_k} d(y_{iv}, c_{kv}) \tag{2.8}$$

Huang et al. [10] extended this method to consider feature weighting per cluster and have shown that feature weighting can be used as an index for feature selection.

Sparsity is a well-documented problem in high-dimensional data. The concept of sub-space clustering has been covered by different authors to deal with this problem [88, 89, 90] where clustering information is retrieved from subspaces of features instead of whole feature space. Jing et al [91] proposed an entropy weighting *k*-Means, *EWk*-Means, to deal with the data sparsity problem in weighted *k*-Means. The *EWk*-Means aims to minimize the within-cluster dispersion and maximize the negative weight entropy by adding weighted entropy in *Wk*-Means objective function. The objective function for *EWk*-Means is:

$$F(S,C,w) = \sum_{k=1}^{K}[\sum_{i\in S_k}\sum_{v\in V} w_{kv}(y_{iv}-c_{kv})^2 + \gamma \sum_{v\in V} w_{kv} log w_{kv}] \tag{2.9}$$

where w_{kv} weight of the entropy of feature v in the cluster S_k is given by

$$w_{kv} = \frac{exp^{\frac{-D_{iv}}{\gamma}}}{\sum_{j\in V} exp^{\frac{-D_{kj}}{\gamma}}} \tag{2.10}$$

and the dispersion of the feature $v \in V$ is the sum of dispersion of entities $y \in S_k$ within the cluster S_k i.e. $D_{kv} = \sum_{i\in S_k}(y_{iv} - c_{kv})^2$.

The Euclidean metric which is used to measure the dissimilarity in *Wk*-Means, originally proposed by Huang et al. [10, 87], is further extended to Minkowski metric by Renato et al. [92]. This algorithm is further discussed in the next chapter.

2.5 Missing Values

The problem of missing values in data mining has been usually studied based on location i.e. where they occur and their relationship with known values. The study of missing values is based on the location usually refers as missing pattern, whereas their study based on the relationship is termed as a missing mechanism. Roderick et al. in their well-known book “Statistical analysis with missing data" [93] have categorized five missing patterns- monotonous, univariate, multivariate, file matching, factor analysis and general, and three missing mechanisms- missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR).

Some of the earlier practices of statisticians to handle missing values are expectation-maximization, EM [94] and multiple imputation [95, 96]. These methods are theoretically correct but their assumption about the distribution of data is unsubstantiated. Moreover, they have very low convergence and their performance is highly influenced by the percentage of missing values [97].

Other common traditional methods of dealing with missing values are:- listwise (or case) deletion, pairwise deletion, and substitution by some values. Listwise (or case) deletion, where the entire data entities or the data features containing the missing values are deleted, is the most common approach and by default is integrated in most of the statistical tools. However, this triggers a huge loss of information [98]. Although pairwise deletion has some advantages over listwise deletion, the correlation matrix that is created to find appropriate pairs has a sub-sampling problem and cannot be reversed [98]. Replacing the missing values by a constant value, like the feature mean [98], mode, class or conceptual mean, or 0, etc. has advantages over the deletion approach as no information is lost. But these methods do not consider feature variation, the relationship between the features and whether the original data can be lost [99].

One of the methods that take the variation of data into account is to replace the missing value with the nearest neighbour imputation, *NN*-impute [100, 101] and its modified variant 'K' nearest neighbour imputation, *kNN*-Impute [102]. *kNN*-impute, find k nearest neighbours of the entities containing that have missing values based on the clean dataset and then replace the missing values with the mean of the feature values of these k neighbours. One of the drawbacks of the *kNN*-impute is that it ignores the correlation between features. To address this problem, Wang and Fu Dong [103] introduced a weighted variant of *kNN*-impute, *WkNN*-impute. The *WkNN*-impute uses a vector regression model to derive weights of the known features with respect to feature with missing values and integrate the feature weights with *kNN*-impute to predict the missing value. The complete case *kNN*-impute(*CCkNN*-impute) only considers the clean (complete) entities to predict missing values. *CCkNN*-impute also has a problem that it becomes ineffective if most of entities have missing values [104] . To address this problem, Jason and Taghi [105] have proposed an incomplete case *kNN*-impute (*ICkNN*-impute) which is also considered the partially missing entities to predict the missing values. *ICkNN*-impute has better performance than *CCkNN*-impute, especially when the dataset contain a large number of missing values.

Olga et al. [106] have used singular value decomposition-based imputation methods, *SVD*-impute, to predict missing values in DNA microarrays. *SVD*-impute derives mutually orthogonal patterns using singular value decomposition. These patterns are then linearly combined to approximate the missing values. Trond et al. [107] have used the least square-based imputation method, *LS*-impute, where correlations between entities are used as a prediction model. *LS*-impute uses the correlation between entities and considers weighted average of several regression estimation to predict the missing values. Local least-square imputation, *LLS*-impute [108], an extension of *LS*-impute, explores the local similarity structure of a dataset. *LLS*-impute selects K similar genes using L_2-norm or Pearson correlation and approximate missing values by least-square optimization. Other popular variants of regression-based imputation models used in different literatures are: Gaussian mixed cluster imputation, *GMC*-impute [109]- data is modelled by Normal distribution and missing values are imputed by EM algorithm; projection onto the convex set, POCS [110]- based on least square estimation but uses constraints/ prior knowledge about the missing values to guide the estimation process; robust least square imputation with principal components, RLSP [111]- addresses the problem of outliers in *LS*-impute by considering l_1-norm instead of l_2-norm or Pearson correlation.

To predict missing feature values, B.M. Patil et al [112] had proposed a new method based on K-Means clustering. The new method is called CMIWD, Clustering Method Imputation with weighted distance which imputes a missing value of an entity as a function of centroid and nearest neighbour. CMIWD works in two phases: first divides the dataset into K clusters and then replaces the missing values. To cluster the data, CMIWD categorizes features into two sets: reference features and non-reference features. The feature that has known values for all entities is called the reference feature and the feature with missing values for at least one entity is called the non-reference feature. A data entity with all reference features is a complete entity. In the first phase, CMIWD divides a dataset into K distinct clusters using K-Means seeded with K centroids randomly chosen from the complete data entities. CMIWD uses a modified distance function, which only considers the reference features to calculate the distance between a centroid and a data entity. The modified distance function helps to assign the incomplete entities to the appropriate cluster. In the second phase, for each missing feature value, CMIWD finds the nearest neighbour among the complete data entities within the same cluster. The distance between the nearest neighbour and the missing entity is calculated based on the reference feature. The distance is called the weight. The missing feature value is then replaced by the mean of the centroid and the weight.

2.6 Lp Norm

Vlachos et al. [113] explored the use of Euclidean distance as a similarity metric for mobile object trajectories, and found that it yielded a large number of outliers. They proposed a flexible sigmoid matching function, which was found to be superior than the L_p metric. However, their work only explored values of p between 0.1 to 2. The research also failed to model the L_p space for the trajectories. Ankita Vimal et al. [114] have used four different distance measures other than Euclidean, and have tested on synthetic datasets generated using Syndeca [115] and one real-world dataset of cricket [116]. They have highlighted the necessary to further extend the work on different datasets using different distance measures and clustering algorithms. Endre Pap et al. [117] derived a generalization of L_p space as a pseudo-L_p space framework and showed the importance of using the L_p space in different clustering applications, like fuzzy systems. Fair *(K,p)*-Means [118], an extension of low-cost fair *k*-means clustering, works under an L_p-norm objective function. The performance of Fair *(K,p)*-means was tested with four real-world datasets - bank, census, creditcard, and census1990 - from UCI machine learning repository [119]. Vincent et al. [120] have derived a theory of approximation of *k*-means and *k*-median for four cases of L_p metrics - L_0, L_1, L_2 and L_∞. However, its optimization is only derived in Euclidean plane. Minkowski metric-based fuzzy Granular classification methods are effective in classification of imbalanced (skewed) datasets [121].The multiple geometric shapes, provided by Minkowski space, are able to extract the key features from imbalanced datasets. These information granular are able to capture important characteristic of data from both majority and minority clusters and hence are effective in classification of imbalance datasets. In the recent work, Vincent et al. [122] have provided a new coreset framework for clustering which simultaneously minimized the closest center problem for (k,p)-clustering problem, where k is the number of centers in p-metric space.

Many datasets, both synthetic and real world, have been used throughout the literature. The measurement of similarities in real-world data are hard to model as the notion of these measures are domain specific. Therefore, to observe the performance of an algorithm in a particular L_p space, researchers have to consider synthetic datasets where the notion of similarity can be controlled. SynDECA [115] - a tool to generate synthetic datasets for cluster evaluation - provides control over the density, radius and number of clusters. The interface can also generate different shape clusters - circular, ellipse, rectangle, square or irregular. However, it does not provide a control over selection of a particular L_p space or notion of similarities. The synthetic data generator interface developed by Pei and Zaiane [123], provides features to control different parameters of clusters such as number of data points, number of clusters, distribution of clusters, density level and complexity of clusters. But, like SynDECA, this tool is also inappropriate for cluster analysis in L_p space as it does not have an interface to control similarity measurement. The GSTD interface [124], a tool kit to generate synthetic datasets based on spatiotemporal information, and IBM's QUEST [125], inherit similar problems and so are not suitable for cluster analysis in L_p space. In a latest literature, Kalke et al. [126] have shown that p-generalized polar method can be used to simulate p-generalized Gaussian distribution.

Chapter 3

Tools, Techniques and Methodologies

In this chapter, different tools, techniques, and methodologies, such as distance metrics, feature weighting, feature selection, *k*-Means variant, etc. which are used in the thesis, are discussed in detail. Different distance metrics used in different literature for similarity measurement in clustering are explained in section 4.1. In section 3.3, two different methods for data standardization are detailed. Feature weighting principle and using them as a feature rescaling factor are explored in section 3.4. Different variants of *k*- ranging from generic to Minkowski variant presented in section 3.5. The mathematical models of clustering validation indexes used during the research are explained in section 3.6. Similarly, section 3.8 has covered the pattern and mechanism of missing values and different techniques used in data mining to addressed them. Finally, techniques used in this thesis for conversion of the dataset in different p-norms are studied in section 4.3

3.1 Similarity Measurement

Clustering refers to the grouping of elements based on their similarity where similar or more “like" items are kept together and dissimilar or more “unlike" items are kept apart. This is achieved by maximizing intra-cluster similarities and minimizing inter-cluster similarities. As clustering algorithms aim to group similar entities together, some sort of measurement of similarity is required. Mathematically, the similarity is measured based on distance metrics. Therefore, the way that similarity in a particular clustering algorithm is defined is crucial in shaping the performance of the algorithm.

In general, two elements x_i and x_j are referred to be similar if they are relatively close, i.e. their distance $d(x_i, x_j)$ is relatively low. Some popular distance metrics used in literature are:

- Manhattan distance, which is also called as city-block or L_1-distance, is the sum of differences between two data points along all the feature spaces $v \in V$ and is given by:

$$d_{Man}(x_i, x_j) = \sum_{v \in V} |x_{iv} - x_{jv}| \tag{3.1}$$

  The median of all data points within a cluster defines a centroid of the cluster when the Manhattan distance matrix is used as similarity measurement. *k*-Means produces diamond-shaped clusters in 2-norms when Manhattan distance is used as a similarity metric.

- Euclidean distance between two data points x_i and x_j is the length of a line segment between the points in the Euclidean space and is defined as:

$$d_{Eud}(x_i, x_j) = \sqrt[2]{\sum_{v \in V} (x_{iv} - x_{jv})^2} \tag{3.2}$$

When Euclidean distance is used as similarity measurement, the centroid of the cluster so formed is given by the mean of all data points within the cluster and the cluster has spherical shape.

- Chebyshev distance, also known as chessboard distance is the maximum difference between two data points along with feature space $v \in V$ and is defined by:

$$d_{Che}(x_i, x_j) = \max_{v \in V} |x_{iv} - x_{jv}| \tag{3.3}$$

Under the Chebyshev distance matrix, the cluster so formed will has a rectangle shape and its centroid is given by $\frac{|x_{iv}+x_{jv}|}{2}$

- The Minkowski distance, the generalization of Euclidean distance and Manhattan distance, between two data points x_i and x_j in L_p norm space is given by:

$$d_{Mink}(x_i, x_j) = \sqrt[p]{\sum_{j=1}^{M} |x_j - x_j|^p} \tag{3.4}$$

where, p is the distance coefficient range from 0 to ∞ which defines the shape of clusters from star to rectangle respectively.

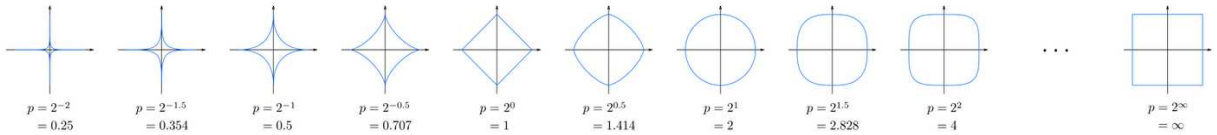


FIGURE 3.1: Different shapes of clusters, in two dimensions, defined by distance coefficient p, are displayed in 2-norm. Each cluster has a spherical shape in the particular p-norm i.e. data points lying on the circumference of the cluster are equidistant from its centroid. This figure is taken from the public domain https://en.wikipedia.org/wiki/Minkowski_distance

Three special cases represented by Minkowski distance, d_{Mink} for different value of p are:
p=1, Manhattan Distance
p=2, Euclidean Distance
$p \to \infty$, Chebyshev Distance

3.2 p-Norms and L_p space

A Norm is a function that maps a real or complex vector space to a non-negative real number in a way that conforms to some properties typical of a distance measure. The mapping describes the length, size or extent of the object from its origin. The most common norm, 2-norm also called the Euclidean distance, defines the square root of the inner product of a vector with itself. The space which is defined by using the particular p-norm is called the L_p space.

3.3 Data Standardization

Data standardization is a basic data-preprocessing step that helps to control variability especially when values of variables lie in a larger range or have greater data variability[127]. Z-score is the traditional method of data standardization where data distribution is translated to zero mean with unit variance and is given by:

$$x_{iv} = \frac{x_{iv} - \mu_v}{\sigma_v} \tag{3.5}$$

Milligan et al. have studied the effect of eight different data standardization methods in cluster recovery and concluded that the range-based methods are better than Z-score to preserve group structure[128]. Range-based methods transform feature values between user-defined ranges; usually between -1 to 1 or 0 to 1. In the range-based method, feature variability is preserved but the range of data dispersion is bounded with a specific range.

$$x_{iv} = \frac{x_{iv} - min_{i=1}^{n} x_{iv}}{max_{i=1}^{n} x_{iv} - min_{i=1}^{n} x_{iv}} \tag{3.6}$$

Schaffer and Green [129] and Stoddard's [130] derived a conclusion that any kind of standardization eliminates between-cluster variability. This conclusion was derived based on UCI datasets [131], true dataset configurations were unknown and hence cannot apply to all datasets [132]. Similar research by Steinley [132] supports Milligan and Cooper's research and found that range standardization has better results in *k*-Means.

In a later development, Mohamad and Usman [127] demonstrated that for the infectious diseases dataset, *k*-Means had better performance with Z-score standardization compared to min-max and decimal scaling, whereas Kamarul et al. [133] found range-based methods were more effective than Z-score standardization in *k*-Means for processing of Malaysia's population and housing census data.

3.4 Feature Weighting

Feature weighting is a technique assigning some values to features, based on their significance. Feature weighting takes into account that different attributes may have different degree of relevance [83, 134, 135] and shows an excellent ability for cluster recovery. The degree of relevance, also called its weighting factor(w_v), is used to rescale the attributes. The weighting factor w_v of a feature v defined in the feature space V = $\{v_1, v_2, ..., v_m\}$ for a dataset Y = $\{y_{iv}\}$ with n entities is given by

$$w_v = \frac{1}{\sum_{u=1}^{|V|} \left[\frac{D_v}{D_u}\right]^{\frac{1}{\beta-1}}} \tag{3.7}$$

where, dispersion of feature v, D_v is an average dispersion of entities y_i from its centroid c_k in cluster S_k, and it is calculated as

$$D_v = \sum_{k=1}^{K} \sum_{i=1}^{|S_k|} (y_{iv} - c_{kv})^2 \tag{3.8}$$

Feature weighting is inversely proportional to the feature variance: lower the variance, higher the weight. In case of cluster based feature weighting, feature weighting factor w_{kv} is given by its relevance within the particular cluster S_k

$$w_{kv} = \frac{1}{\sum_{u=1}^{|V|} \left[\frac{D_{kv}}{D_{ku}}\right]^{\frac{1}{\beta-1}}} \tag{3.9}$$

where, the dispersion of feature v in cluster S_k, D_{kv} is

$$D_{kv} = \sum_{i=1}^{|S_k|} (y_{iv} - c_{kv})^2 \tag{3.10}$$

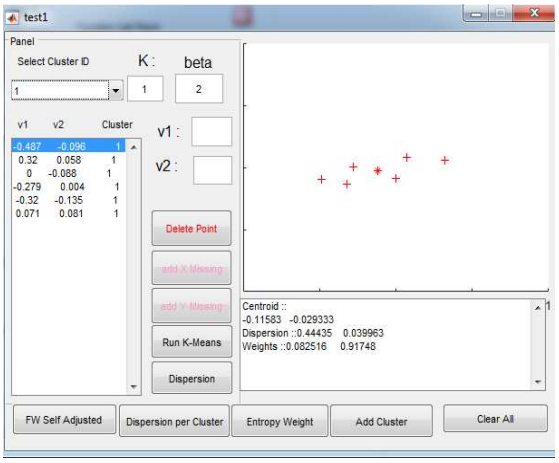


FIGURE 3.2: A cluster of data points which have higher dispersion along x-axis than y-axis is created using the GUI where + and * represent data points and centroids respectively.

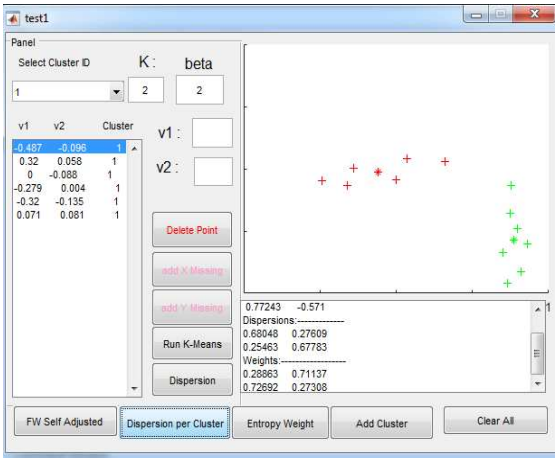


FIGURE 3.3: Two different clusters- red and green are created using the GUI. Red cluster has data points with higher dispersion along x-axis whereas green cluster has data points with higher dispersion along y-axis.

MATLAB based GUI, shown in the figure above, is created to study the relationship between feature dispersion and their weight with respect to the whole dataset and clusters. This GUI allows a user to enter data points and create multiple clusters in two dimensions and calculate their centroids, dispersion, and weights.

Example 1: Feature weighting in whole dataset

In Figure 3.2, using the designed GUI, a dataset is created by adding six data points. This dataset is defined in feature space $V = \{v_1, v_2\}$ corresponding to the x-axis and y-axis. The dispersion of the data set along the x-axis d_{v_1} and y-axis d_{v_2} are 0.4443 and 0.0400 respectively. The result is quite obvious as entities in the dataset are highly dispersed along the x-axis compared to the y-axis. Feature weights for the dataset which are calculated by GUI using equation 3.7 are 0.0825 and 0.9175 for v_1 and v_2 respectively. Hence, it reflects the inverse relation between feature dispersion and their weights.

Example 2: Feature weights in two clusters

In figure 3.3, a dataset with two clusters, red and green, is created using the GUI. Data points in the red cluster have higher dispersion along the x-axis than the y-axis whereas, in the green cluster data points are more scattered along th e y-axis than the x-axis. The dispersion of data points along the x-axis in the red cluster, $D_{red,x-axis} = 0.680$ is greater than the dispersion of data points along the x-axis in the green cluster, $D_{green,x-axis} = 0.254$ and vice-versa in the y-axis, which shows that dispersion of features has influenced their weight.

In the GUI, feature weighting along the x-axis and y-axis are 0.5100 and 0.4900 overall. Data points are equally dispersed along both axes when the whole dataset is considered, which is reflected in their weights.

3.5 *k*-Means Algorithms and its Weighting Variant

3.5.1 Generic *k*-Means

k-Means [20, 19] is the most popular and simple partition-based clustering algorithm. Assume that a dataset Y has cardinality n i.e. $|Y|$= n and is defined in a feature space V = $\{v_1, v_2, ..., v_m\}$. *k*-Means divides the dataset Y into K disjoint sets $S = \{S_1, S_2, ...S_K\}$, where $\bigcup_{i=1}^{K} S_i = Y$ and $S_i \cap S_j = \emptyset \; \forall i \neq j$. Each of these disjoint sets S_k is a cluster and is uniquely represented by a data point called a centroid, c_k. The objective of *k*-Means is to minimize the sum of dissimilarity, F(k) between each entity $y_i \in S_k$ and its centroid $c_k, \forall k = 1, 2, ...K$.

$$F(k, C) = \sum_{k=1}^{K} \sum_{y_i \in S_k} d(y_i, c_k) \tag{3.11}$$

where, C = $c_1, c_2, ..., c_K$ is the set of all centroids and d is the distance function which measures the dissimilarity between the entity y_i and its centroid c_k. In generic *k*-Means squared Euclidean distance is used as a dissimilarity measure.

$$d(y_i, c_k) = \sum_{v \in V} (y_{iv} - c_{kv})^2 \tag{3.12}$$

Centroid c_k is the centre of the gravity of cluster S_k and is given by:

$$c_k = \frac{\sum_{i=1}^{|S_k|} y_i}{|S_k|} \tag{3.13}$$

Therefore the optimization in 3.11 can be written as

$$F(k, C) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} (y_{iv} - c_{kv})^2 \tag{3.14}$$

The *k*-Means criterion in equation (3.14) follows three steps for optimization:

1. start with an initial set of centroids C = $\{c_1, c_2, ..., c_k\}$
2. assign each entity $y_i \in Y$ to its corresponding cluster S_k
3. update each centroid c_k to its centre of gravity as given by equation 3.13,
4. go to step 2 until convergence.

The complexity of *k*-Means is defined by $O(n|V|Kt)$, where n is the cardinality of dataset Y, $|V|$ is the number of features, t is the number of recursive steps *k*-Means takes to converge and K is the number of clusters.

3.5.2 *ik*-Means

Intelligent *k*-Means, *ik*-Means is the deterministic version of *ik*-Means where initial centroids are chosen using so-called anomalous pattern (AP). The anomalous pattern is a recursive process of partitioning the dataset into two subsets where the first subset contains points near the grand mean (called the reference point) and the next subset contains the data point near to the farthest point from the reference point. The process is recursive on the first sub-set until a stop criterion

is matched. For details see [11] chapter 3.

Algorithm *ik*-Means
Input:Y dataset, θ minimum number of elements in a cluster
Output: C centroids

1. let t=1,
2. Y^t = Y.
3. Set reference point $a = \bar{Y^t}$, the center of dataset Y^t.
4. Find a point in Y^t which is farthest from the reference point a, $f^t = \max_{y_i \in Y^t} d(y_i, a)$
5. Fixing one of the centroids a, Run *k*- for K=2 on Y^t with $C^t = a, f^t$ as initial centroids such only f^t keep updating until the cluster S_f is anomalous in relation to a. This will yield two clusters S_a and S_f corresponding to centroids a and f^t
6. if $|S_t| \geq \theta$ add f^t to C
7. set $Y^t = Y^t - S_f$, removing elements of cluster S_f from Y^t.
8. if $|Y^t| \geq 1$, set t=t+1 and got to step 3.
9. Run *k*- with C and $K = |C|$

Apart from adding deterministic to *k*-Means, *ik*-Means is used to predict cluster tendency and is successfully applied in different comparative experiments [136].

3.5.3 *Wk*-Means

The weighted version of *k*-, *Wk*-Means, takes into account that different attributes may have different degrees of relevance [83, 134] and shows an excellent ability for cluster recovery. The degree of relevance also called the weighting factor (w_{kv}), is used to rescale the attributes and is given by equation 3.9. The distance function in generic *k*- 3.11 is modified by weighted distance

$$d_w(y_i, c_k) = \sum_{v \in V} w_{k,v}^{\beta} (y_{iv} - c_{kv})^2 \tag{3.15}$$

where, the attribute weighting factor $w_{k,v}$, given by equation 3.9, is put to the power of the exponent β to avoid linearity. The centroids in 3.13 modified as

$$c_{kv} = \frac{\sum_{i=1}^{|S_k|} w_{k,v} * y_{i,v}}{|S_k|} \tag{3.16}$$

The objective function in equation 3.14 is modified as

$$F(K, C, w) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} w_{k,v}^{\beta} (y_{iv} - c_{kv})^2 \tag{3.17}$$

3.5.4 *MWk*-Means

Minkowski Weighted *k*-Means (*MWK*-Means) [92, 137] modifies the objective in equation 3.17 where the feature weighting factor β is used as distance coefficient to ensure feature weighting is used as feature rescaling factor. The distance function in 3.15 is redefined as

$$d_{w\beta}(y_i, c_k) = \sum_{v \in V} w_{k,v}^{\beta} |y_{iv} - c_{kv}|^{\beta} \tag{3.18}$$

Using the equation , the criterion of generic *k*-Means is modified to obtain criterion for *MWk*-Means

$$F(k, C, w, \beta) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} w_{k,v}^{\beta} |y_{iv} - c_{kv}|^{\beta} \tag{3.19}$$

The weighting factor w_{kv}, calculated using equation 3.9, ensures that attributes are treated on the based of their variance: attributes uniformly distributed across clusters get a smaller weight while those concentrated around cluster get a larger weight [ref]. Apart from attribute weighting, *MWk*-Means nullifies *k*-Means bias towards a spherical shape by redefining the dissimilarity measurement using the Minkowski distance metric.

The iterative optimization is similar to the optimization of the *k*-Means criterion except for an extra step of calculation of feature weighting is added and the centroid is replaced by the Minkowski center.

3.6 Cluster Validity

Cluster validation is an important step in clustering which evaluates the clustering structure detected by a clustering algorithm. It helps to measure how close the clustering matches with the internal structure present in the data. This process is complicated as there is no correct label (unsupervised learning) available, the number of "correct" clusters is unknown in most cases, cluster structure varies between clustering algorithms and the dataset may have a high level of noise or "true" random features.

Jain and Dubes [138] have categorized the clustering validation indices into three groups: internal, external and relative. Internal indices use the information intrinsic in a dataset to evaluate a clustering structure derived by a clustering algorithm. None of the label information, only data (i.e. purely unsupervised) is used during the validation process. Whereas, external indices evaluate the clustering results with respect to pre-specified clusters. Internal indices are usually used to turn the parameters within the clustering algorithm whereas external indices are used to validate the performance of the clustering algorithm on the “ground truth".

In this research, Silhouette and Calinkski-Harabasz indices are used as internal indices to automate the selection of parameters in the proposed algorithms. Whereas, adjusted Rand Index (ARI) is used as an external index to evaluate the performance of the algorithms considered in the research by comparing them with the actual clusters.

- **Silhouette Index**
  Silhouette index is a graphical aid method that is developed for the interpretation and validation of cluster analysis [139]. Silhouette index is found to be effective in many clustering algorithms where the number of cluster are not known [140, 11]. The Silhouette coefficient combines both cohesion and separation, and is given by

  $$s(y_i) = \frac{b(y_i) - a(y_i)}{max(a(y_i), b(y_i))} \tag{3.20}$$

  where:

  $a(y_i)$ is the average dissimilarity of entity $y_i \in S_k$ and $y_j : y_j \in S_k, y_i \neq y_j$ and

$b(y_i)$ is the lowest average dissimilarity between y_i and $y_j : y_j \in S_l, y_i \in S_k$ for $l \neq k$.

- **Calinski-Harabasz Index**
  Calinski-Harabasz Index, commonly known as CH Index, is a "variance ratio criterion" for evaluating the clusters in a multi-dimensional Euclidean space [141]. CH uses ratio of between-cluster means (data scatter) and within cluster's sum of square error (SSE).

$$CH = \frac{B}{W} X \frac{N-K}{K-1} \tag{3.21}$$

  where,

  W is the overall within-cluster variance,

  B is the overall between-cluster variance,

  N is the number of entities and,

  K is the number of clusters.

- **Adjusted Rand Index**
  Adjusted Rand index, ARI is an external validation index that compares the resulted cluster with the "ground truth" (known labels) using a confusion matrix. For supervised classification, ARI is found to be effective for performance measurement and feature selection [142].

$$ARI = \frac{yy + nn}{N_T} \tag{3.22}$$

  where,
  yy, nn and N_T is defined by the confusion matrix between two partition P_1 and P_2
  $N_T = \frac{N(N-1)}{2} = yy + yn + ny + nn$, is the total number of pair between P_1 and P_2
  yy = true positive, yn = true negative, ny = false positive and nn = false negative.

3.7 Feature Selection

In this section, four different feature selection algorithms: intelligent *k*-Means feature selection (ikFS), feature selection using feature similarity (FSFS), multi-cluster feature selection (MCFS), and feature selection via feature weighting (FSFW) are discussed in detail.

3.7.1 intelligent K-Means Feature Selection(ikFS)

Intelligent *Wk*-Means for feature selection(ikFS) [25, 143] aims to cluster features based on their similarity. Instead of clustering entities, ikFS clusters similar features together and then identifies and removes features that are redundant. During the process, two major factors ikFS has to address are: (i) identifying the number of clusters of features; and (ii) number of features that can be selected from a particular cluster. Issue (i) is solved using *ik*-Means, which not only finds the appropriate number of clusters of features but also finds the best initial centroids for *k*-Means. Selecting most informative features from each cluster of features looks obvious solution for (ii) and the number of features to be selected in a particular cluster is given by

$$F_k = \frac{|S_k|}{|V|} K \tag{3.23}$$

Where, $|S_K|$ is the cardinality of a given cluster of features, $|V|$ is the number of features of the dataset Y and K is the number of clusters of features. F_k ensures that number of features to be

selected from a cluster depends on the number of features in the particular cluster. After clustering of features into K clusters, F_k are chosen from S_k based on feature selection using feature similarity(FSFS) index.

ikFS Algorithm

- transport a given dataset Y so that orginal features become entities.
- standardise the dataset
- find K clusters of features using *ik*-Means
- from each cluster S_k, select F_k features using FSFS.

3.7.2 Feature Selection using Feature Similarity

Feature selection using feature similarity (FSFS) [23] is one of the most cited unsupervised feature selection algorithms. It calculates pairwise feature similarities in order to determine a set of maximally independent features, and then discards those that are considered redundant. This is the first algorithm in history to use the maximum information compression index, defined below, in a feature selection scenario.

$$2\lambda_2(v_t, v_j) = var(v_t) + var(v_j) - \sqrt{(var(v_t) + var(v_j)^2 - 4var(v_t)var(v_j)(1 - \sigma(v_t, v_j)^2)}, \tag{3.24}$$

Where, $\sigma(v_t, v_j)$ denotes the Pearson correlation coefficient between v_t and v_j, and $var()$ represents the variance of a feature pass as a parameter. The value of λ_2 is zero when the features are linearly dependent, and increases its value as the amount of dependency decreases.

Symbols used in algorithms above,

- $m_{V'} = |V'|$ is the cardinality of V'
- k is the number of nearest neighbours (provided as parameter)
- $r^k_{v'}$ is dissimilarity between feature v'_i and its k^{th} nearest features which is calculated using the equation(3.24).

Feature selection using feature similarity (FSFS)

1. select a value for k, subject to $k \leq m - 1$.
2. set $V' \leftarrow V$.
3. for each $v' \in V'$, compute $r^k_{v'}$.
4. find the feature v'^* for which $r^k_{v'^*}$ is minimum.
5. remove from V' the k nearest-neighbours of v'^*.
6. set $\epsilon = r^k_{v'^*}$.
7. if $k > m_{V'}$ go to Step 9.
8. while $r^k_{v'^*} > \epsilon$
   (a) $k = k - 1$.
   $r^k_{v'^*} = \inf_{v' \in V'}$
   (k is decremented by 1, until the kth nearest-neighbour of at least one of the features in V' is less than ϵ-dissimilar with the feature).

(b) if $k = 1$ go to step 9.

9. go to Step 2.

10. return the feature set V' as the reduced feature set.

3.7.3 Multi-Cluster Feature Selection

Ranked based feature selection algorithms like FSFS have reduced the computational cost of finding optimal sets of “true" features, as calculation of scores for each features is independent of each other. However, since the possible correlations between features are neglected, ranked based feature selection algorithms can not guarantee the selection of an optimal feature subset. Inspired by recent developments in spectral analysis (manifold learning) [144, 145] and L1-regularized models for subset selection [146, 147] Cai et al. [24] introduced Multi-Cluster Feature Selection (MCFS), which uses multiple eigenvectors of graph Laplacian matrix for selection of “true" features so that multi-cluster structure of the data is preserved.

Multi-cluster feature selection

Inputs:
K- number of clusters;
d-number of selected features;
p- number of nearest neighbours;

Steps

1. Construct a k-nearest-neighbours graph based on spectral clustering.
2. Solve a generalised eigen-problem, leading to the K top eigenvectors with respect to the smallest eigenvalues*.
3. Solve K L1-regularised regression problems, leading to K sparse coefficient vectors.
4. Compute the MCFS score for each feature.
5. Return the top $m_{V'}$ features according to their MCFS scores.

Details of each steps from 1-4 are discussed in the original paper [24].

Note*: smallest eigenvalues are chosen in order to maximize the convergence rate in symmetric metric [148].

3.8 Missing Data

Missing data pattern and missing data mechanism are two fundamental issues that help us to understand missing values. Missing data pattern observes the location of missing values within a dataset, whereas missing data mechanism derives the relationship between the missing values and known values. In this section, the theoretical background of these two approaches and different methods used to address missing values are explained in detail.

3.8.1 Missing Data Patterns

Missing data patterns indicate the occurrence of missing values. Missing data patterns can be categorized as univariate and multivariate based on the number of attributes possessing missing values. In univariate, only one attribute has missing values but in multivariate, there is more

than one attribute with missing values.

Missing-data indicator matrix M = $\{m_{iv}, i = 1, 2, ..., n \text{ and } v \in V\}$ is introduced here to represent missing data where

$$m_{iv} = \begin{cases} 1, & \text{if attribute value } y_{iv} \text{ is missing} \\ 0, & \text{otherwise} \end{cases}$$

It is helpful to sort rows and columns of the given data according to the pattern of missing data to observe missing data patterns. Entities with missing values are pushed down the table and the attributes possessing missing values are moved towards the rightmost end of the data table. Figure 3.4 shows some of the major missing data patterns identified after sorting.

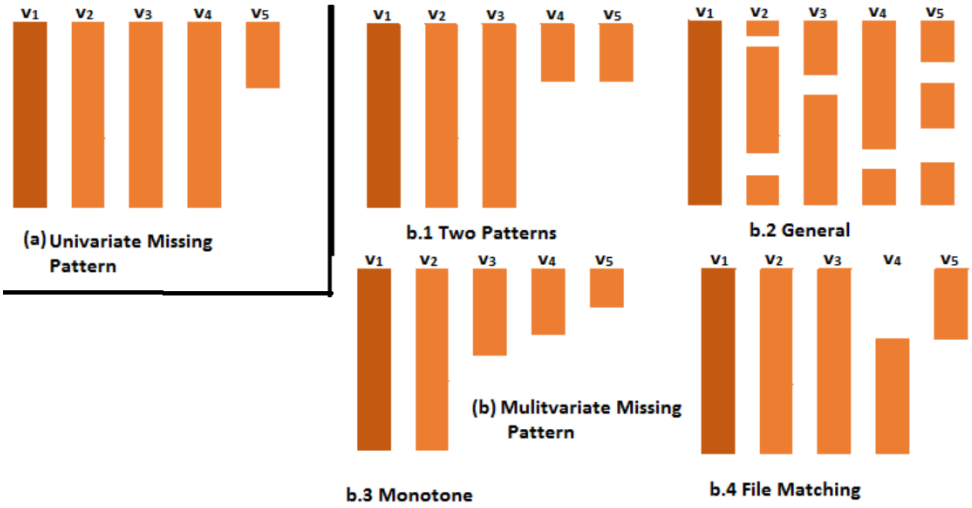


FIGURE 3.4: Examples of missing-data patterns. Rows and columns correspond to observations and variables respectively. Data rows and columns are sorted according to the pattern of missing data.

1. Univariate Missing Pattern
   As illustrated in figure 3.4(a) presence of missing values in univariate missing pattern is restricted to only one attribute (rightmost attribute in sorted data table). This is also referred to as a missing-plot problem where the dependent attribute v_m is plotted from given m-1 known attributes, $V^{known} = \{v_1, v_2, ..., v_{m-1}\}$. For example, in a survey, there is likely a trend where all the non-response individuals fail to answer the same question.

2. Multivariate Missing Pattern
   In multivariate missing pattern there is more than one attribute with missing values. This pattern can be generalized to a missing-plot problem as:

   - out of m set of attributes, we have at most m-2 known attributes, $V^{known} = \{v_1, v_2, ..., v_{m-2}\}$
   - there are at least two dependent attributes, v_{m-1} and v_m
   - some of the functional values are missing in the dependent attributes

   Multivariate missing pattern can be further categorized as:

   (a) Multivariate n-patterns
   As shown in figure 3.4(b.1), in multivariate n-pattern $|V^{missing}| \geq 2$ with a constraint $m_{i,v_a} = 1 \implies m_{i,v_b} = 1, \forall v_a, v_b \in V^{missing}$.

   (b) Monotone Multivariate
   In monotone multivariate missing pattern, missing attributes can be sorted in an order such that an entity having missing values in a high order attribute (leftmost)

and also has missing values in all attributes below the higher order (rightmost) i.e. $m_{i,v_a} = 1 \implies m_{i,v_b} = 1, \forall v_a, v_b \in V^{missing}$ and $a \leq b$ as shown in figure 3.4(b.3).

(c) File Matching
One of most common reason of introducing missing values are adding or matching data files(dataset) with some features which are not common among them. For example, in the figure 3.4(b.4), two dataset Y^1 and Y^2 are combined where Y^1 is defined in feature space $V^1 = \{v_1, v_2, v_3, v_4\}$ and Y^2 is defined in feature space $V^2 = \{v_1, v_2, v_3, v_5\}$. In the combined dataset $Y = \{Y^1 \cup Y^2\}$ all entities from Y^1 have missing values for feature v_5 and all entities from Y^2 have all value missing values for feature v_4.

(d) General
If correlation of occurrence of missing values among $V^{missing}$ features is not found then it is categorized as general missing pattern.

3.8.2 Missing-data Mechanism

In the previous section, various patterns of missing data were observed. Another important aspect on the study of missing values is to find the mechanism that lead to missing data. Missing data mechanism is characterized by a conditional probability $f(M\|Y, \theta)$ for some unknown parameter θ where Y^{known} is a complete dataset and $M = m_{iv}$ is the missing data indicator.

	v1	v2	v3
	Age	Gender	Salary
	20	M	3400
	25	M	2000
	30	M	3100
	32	M	4500
	21	F	2000
	34	F	3000
	67	F	2500

(a) Original Dataset Y

	v1	v2	v3
	Age	Gender	Salary
	20	M	3400
	25	M	
	30	M	
	32	M	4500
	21	F	2000
	34	F	
	67	F	2500

(b) Missing Completely At Random

	v1	v2	v3
	Age	Gender	Salary
	20	M	3400
	25	M	
	30	M	3100
	32	M	
	21	F	2000
	34	F	3000
	67	F	2500

(c) Missing At Random

	v1	v2	v3
	Age	Gender	Salary
	20	M	3400
	25	M	2000
	30	M	3100
	32	M	
	21	F	2000
	34	F	
	67	F	2500

(d) Not Missing at Random

FIGURE 3.5: Examples of missing data mechanism.

1. Missing Completely at Random (MCAR)
   If the conditional probability f is independent of Y, it is called missing completely at random (MCAR). For example, in Figure 3.5(b), some of the values for the attribute salary are missing randomly i.e. the missingness in salary is independent of values in other attributes or itself.

2. Missing at Random (MAR)
   In missing at random (MAR), missingness is random within itself but range of its occurrence is defined by other independent attributes. For example, in the figure 3.5(c) missingness in the attribute salary depends on the values in the attribute gender as only two of the entities missing salary belong to 'male' gender.

3. Not Missing at Random (NMAR)
   In not missing at random (NMAR), missingness is a function of values in the attribute itself. For example, in Figure 3.5(d), all missing values for salary are greater than 2500.

3.8.3 Dealing with Missing Attribute Values

A common solution for missing attribute values is to remove the problematic parts of data i.e. removing entities (row-wise deletion) or attributes (column-wise deletion) whose values are missing. This leads to loss of information, hence it is not recommended. There are two preferred solutions for missing values. The first one is replacing the values with some known values which is called an imputation method. Another approach is, re-writing data mining algorithms so that the algorithm either discard missing values or learn somehow to replace missing values based on the data distribution.

Imputation Methods

In this section, common imputation methods like attribute mean value, kNN imputation, regression based imputation etc. are discussed.

Attribute Mean Value

One of the easiest approaches to deal with missing values is to replace them by their attribute-mean, mode, or median calculated over the whole dataset or within the cluster (concept). The conceptual method requires prior knowledge about clustering (levels). When no clustering information is provided, conceptual imputation can still be carried out but requires the selection of an appropriate clustering algorithm to recover clusters before imputation.

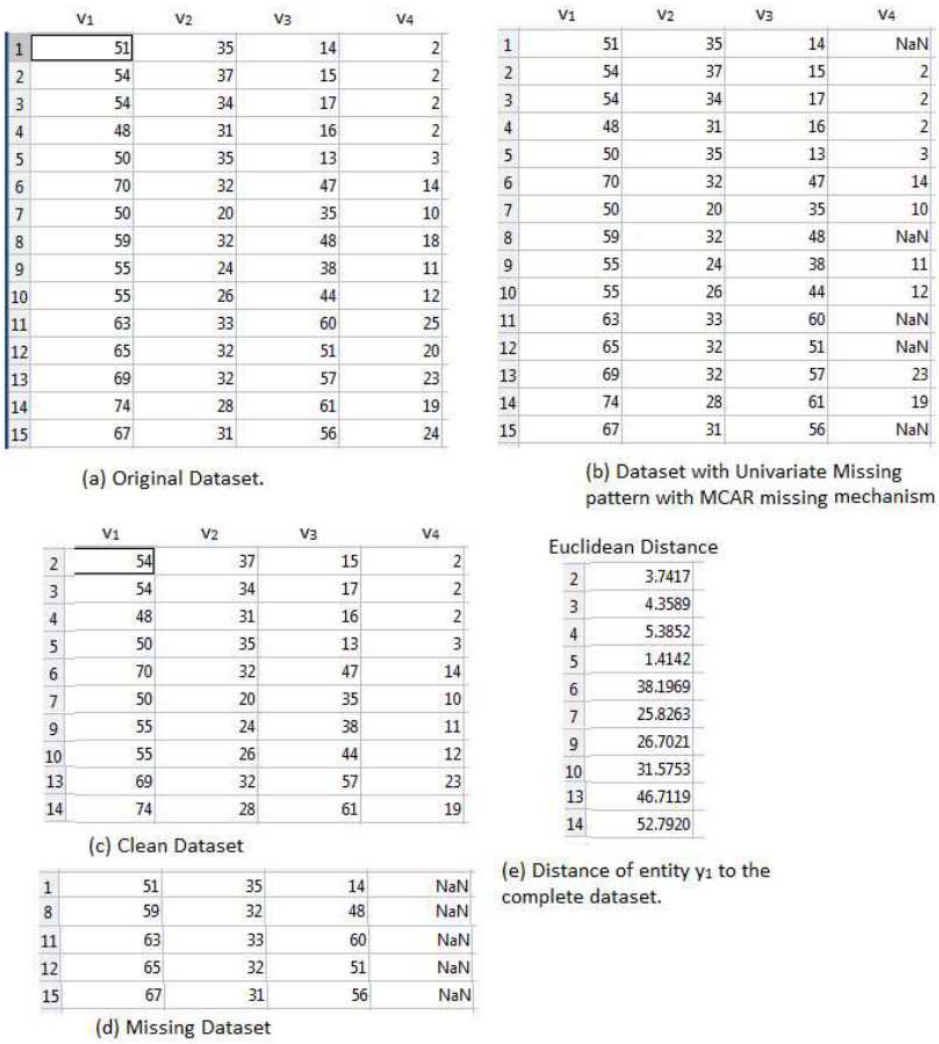
k-Nearest Neighbour Techniques

The *k*-Nearest Neighbour (kNN) technique is based on the nearest neighbours first concept. kNN based imputation algorithms find *K* nearest neighbours based on some similarity function, and then impute missing values with the corresponding attribute mean of *K* nearest neighbours. Depending on the selection of neighbours, kNN based imputation algorithms are further categorized as Complete Case kNN (*CC*-kNN) and Incomplete Case kNN (*IC*-kNN). In *CC*-kNN imputation, neighbours are selected only from clean or complete dataset whereas in *IC*-kNN entities with missing attribute values are also considered as neighbours.

1. **Complete Case kNN (*CC-kNN*)**

   The Complete Case kNN only considers entities with all known values as a possible candidate for nearest neighbours selection. Therefore, **CC**-kNN divides the dataset into two

which are- clean dataset and missing dataset. Clean dataset which is also called an observed dataset, consists of entities with all known values, whereas missing dataset consists of entities with at least one missing value. Missing values in missing dataset are replaced with the mean of K nearest neighbours from the clean dataset.



	V1	V2	V3	V4
1	51	35	14	2
2	54	37	15	2
3	54	34	17	2
4	48	31	16	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
8	59	32	48	18
9	55	24	38	11
10	55	26	44	12
11	63	33	60	25
12	65	32	51	20
13	69	32	57	23
14	74	28	61	19
15	67	31	56	24

(a) Original Dataset.

	V1	V2	V3	V4
1	51	35	14	NaN
2	54	37	15	2
3	54	34	17	2
4	48	31	16	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
8	59	32	48	NaN
9	55	24	38	11
10	55	26	44	12
11	63	33	60	NaN
12	65	32	51	NaN
13	69	32	57	23
14	74	28	61	19
15	67	31	56	NaN

(b) Dataset with Univariate Missing pattern with MCAR missing mechanism

	V1	V2	V3	V4
2	54	37	15	2
3	54	34	17	2
4	48	31	16	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
9	55	24	38	11
10	55	26	44	12
13	69	32	57	23
14	74	28	61	19

(c) Clean Dataset

	V1	V2	V3	V4
1	51	35	14	NaN
8	59	32	48	NaN
11	63	33	60	NaN
12	65	32	51	NaN
15	67	31	56	NaN

(d) Missing Dataset

Euclidean Distance

2	3.7417
3	4.3589
4	5.3852
5	1.4142
6	38.1969
7	25.8263
9	26.7021
10	31.5753
13	46.7119
14	52.7920

(e) Distance of entity y_1 to the complete dataset.

FIGURE 3.6: An example set for complete case kNN (*CC*-kNN) imputation in a dataset with univariate missing pattern and MCAR missing mechanism to impute missing values in entity y_1

.

In the figure 3.6(b) dataset is first divided into clean dataset [figure 3.6(c)] and missing dataset [figure 3.6(d)]. To replace the missing values $y_{1,4}$ in entity y_1, dataset Y is first divided into clean and missing dataset. Then entities in the clean dataset are sorted by their distance to y_1 [figure 3.6(e)] and for k equals to three (given), three nearest neighbours: $\{y_5, y_2, y_3\}$ are chosen. And finally, the missing value $y_{1,4}$ is imputed by the attribute mean of three neighbours. Here $y_{1,4}$ is replace by $\frac{y_{5,4}, y_{2,4}, y_{3,4}}{3} \rightarrow \frac{3+2+2}{3} \rightarrow 2.33$.

2. **Incomplete Case kNN (*IC*-kNN)**

   Unlike complete case kNN, *IC*-kNN considers all entities as the possible candidates of nearest neighbours. In the case of univariate missing pattern, the selection of nearest neighbours is based on only observed attributes. For multivariate missing pattern, observed attributes between two entities are considered for similarity measurement. Since

the number of missing attributes between entities varies, in multivariate missing pattern, the similarity measurement is normalized to nullify the biasedness created by number of missing attributes. The normalization is achieved by dividing the similarity measurement with the number of observed attributes between entities.

For each missing entity $y_i \in Y$ (dataset)

(a) Let O be the set of observed attributes in y_i

(b) For each missing attribute v_j in y_i

i. Let $T \subset Y | \forall t \in T, t_{O \cup j}$ is observed.

ii. Let $C \subset T$ is K nearest neighbours of x_i drawn from T based on observed attributes O.

iii. Set $y_{ij} = \frac{\sum_{c \in C} c_j}{K}$

	V1	V2	V3	V4
1	51	35	NaN	2
2	54	37	15	NaN
3	54	34	17	2
4	48	31	NaN	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
8	59	32	48	18
9	55	24	38	11
10	55	26	44	12
11	63	33	NaN	NaN
12	65	32	51	20
13	69	32	57	23
14	74	28	61	NaN
15	67	31	56	24

(a) Mulivariate Missing pattern with MCAR mechanism

	V1	V2	V3	V4
3	54	34	17	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
8	59	32	48	18
9	55	24	38	11
10	55	26	44	12
12	65	32	51	20
13	69	32	57	23
15	67	31	56	24

(b) Complete Case Dataset

	V1	V2	V3	V4
2	54	37	15	NaN
3	54	34	17	2
5	50	35	13	3
6	70	32	47	14
7	50	20	35	10
8	59	32	48	18
9	55	24	38	11
10	55	26	44	12
12	65	32	51	20
13	69	32	57	23
14	74	28	61	NaN
15	67	31	56	24

(c) Incomplete Case Dataset with respect to y_1 entity.

FIGURE 3.7: Example of incomplete case kNN, *i*C-kNN in a dataset with univariate missing pattern with MCAR missing mechanism.

Using the dataset from figure 3.6(a), new dataset with missing values is created so that its missing values have multi-variate pattern and MCAR mechanism. One of such instance is defined in the figure 3.7(a). The complete case analysis of the missing dataset has only 10 elements in a clean dataset. However, there are 12 entities that can be used for imputation of missing element in y_1 for an incomplete case.

Regression Based Imputation

Regression based imputation methods, an alternative to mean imputation, predict the missing values based on relation between the observed and missing values. For example, salary of a customer in a banking dataset may depend on the age and job status. Based on the observed values for similar age and job group, the missing (if any) salary of an employee can be predicted. There are conceptual and non-conceptual regression based imputation method like kNN imputation.

1. **Simple Regression**

   Simple regression (non-conceptual) method divides a dataset into a clean dataset and a missing dataset so that a clean dataset contains only observed values. For each attribute from the missing dataset, a regression model is formed based on the clean dataset. Missing values are then imputed using these regression models.

2. **Regression with Cluster**

   Clustering properties are likely to be preserved in dataset. For example, customer database of a bank can be grouped based on the attributes like age, gender income or occupation. Shin et al. [149] use this clustering tendency for estimating the missing values by integrating clustering and regression techniques together. The proposed method RegressionCluster (RC) [150] first estimates missing values using regression statistic. Once a clean dataset is recovered, clustered are retrieved from the dataset. Missing values are again recalculated using the regression statistic within the cluster.

   RegressionCluster(RC)

   (a) divide the dataset Y into Y^{clean} and $Y^{missing}$ where $Y^{missing}$ has all entities from Y with at least one missing attribute value and Y^{clean} = Y - $Y^{missing}$.

   (b) using Y^{clean} as a base, recover the missing values in $Y^{missing}$ using regression method. Let Y′ be the new imputed dataset.

   (c) recovery K clusters $S_1, S_2, ..., S_k$ from Y′ using appropriate clustering algorithm.

   (d) for each cluster S_i, the regression analysis in step 1,2 and 3 is applied for imputation of missing values $y_{ij} \in Y^{missing} \cap y_{i,j}$ using the base $Y^{clean} \cap y_{i,j}$.

No Imputation

Imputation method provides an easy solution for missing values. However, replacing original values may lose the diversity of the dataset, hence yield poor data analysis. For example, replacing missing values with attributes mean drives the data distribution towards the predicted mean.

One of the alternatives of imputation method is to modify algorithms which can handle missing values implicitly. Some authors replace similarity measurements (distance function) to deal only with the known values called partial distance. Soft constraint techniques are also used to penalize the entities with missing values in order to nullify their effect.

In the section below, use of partial distance in different *k*-Means type algorithms to cluster dataset with missing values is discussed.

Partial Distance

Partial distance between two entities calculates the distance by considering only the known attributes among them. The distance is then normalized to compensate for the missing ones.

Partial distance between two entities y_i and y_j with mF number of missing attributes and oF number of observed (known) attributes is given by:

$$PD(y_i, y_j) = \frac{oF + mF}{oF} \sqrt{\sum_{v \in V} (Dist_v)^2} \tag{3.25}$$

where,
$Dist_v$ is the similarity of two entities y_i and y_j in v attribute

$$Dist_v = \begin{cases} 0 & y_{i,v} \text{ or } y_{j,v} \text{ is missing} \\ (y_{i,v} - y_{j,v}) & \text{otherwise} \end{cases} \tag{3.26}$$

Partial distance assigns the average distance of the known attributes to each missing attributes. For example, the partial distance between two vectors [3 nan 4 5 6] and [5 6 nan 7 8] is calculated as: $\frac{5}{3}\sqrt{(3-5)^2 + (5-7)^2 + (6-8)^2}$

In the example above, nan denotes missing values. Distance between two vectors is calculated using the first, fourth and fifth elements of both vectors. Distance so obtained is then normalized by the ratio of number of known attributes between two vectors over the total number of attributes.

k-Means with partial distance (*k*-Means*PD*)

k-Means with partial distance [151], *k*-MeansPD, replaces the Euclidean distance which is used to calculate the similarity between centroids and data points by partial distance [equation(3.25)] to cope with missing values. This approach cuts off the initial data pre-processing step that required the imputation of missing values before *k*-Means is applied. *k*-MeansPD follows the same basic steps of *k*-Means: starts with *K* random centroids and iteratively updates centroids to the center of the gravity of the clusters until it reaches predefined termination conditions. *k*-MeansPD has three major modifications to the generic *k*-Means : initialization of centroids, measurement of similarity between the missing data entities and centroids, and redefining centroids in each iteration.
k-Means*PD* divides a dataset into two sets: clean dataset and missing dataset. Initial *K* centroids are chosen from the clean dataset to avoid missing values. In each iteration, centroids are replaced by respective clusters' mean where only the known values are considered.

$$c_{kv} = \sum_{i=1}^{N_k} o_{iv} * y_{iv} * \frac{|S_k|}{nO_v}, \tag{3.27}$$

where, nO_V is total number of known attribute values for attribute v in cluster S_K, given by

$$nO_v = \sum_{i=1}^{N_k} o_{iv} \tag{3.28}$$

And,

$$o_{iv} = \begin{cases} 1, & \text{if } y_{iv} \text{ is known value} \\ 0, & \text{otherwise} \end{cases}$$

3.9 Dataset

To evaluate the performance of the proposed algorithms, synthetic datasets from the Gaussian mixed model (GMM), under control parameters and real-world datasets from the UCI repository are used in this thesis. In addition to this, to observe the resistance of the algorithms against

noise, three different types of noise based on their distribution, model and type are added in the UCI and synthetic datasets.

3.9.1 Synthetic Gaussian Distribution

The distribution of data in a real-world dataset is hard to identify. Therefore, to establish a theoretical understanding of clustering algorithms experiments are conducted with synthetic datasets generated under a controlled environment. The Gaussian mixed model is a popular tool used by many researchers to generate clusters [152, 153]. The aim of this section is to introduce different parameters considered while defining the Gaussian mixed model for synthetic dataset

For standard scenarios, we have generated datasets with a mixed-model Gaussian distribution. The distribution consists of 12 different configurations: three types of noisy features and four types of feature space cardinality. The three types of noise are no noise, half noise, and full noise, and four different cardinalities of the features space: 8, 12, 16, and 20. The distribution has spherical Gaussian clusters with diagonal covariance matrices of $\sigma^2 = 0.5$. To remove biases we have considered 20 different datasets for each of the four configurations.

TABLE 3.1: The experiment of synthetic mixed-model Gaussian distribution contains 20 different datasets with 1000 data instance in each. The distribution has spherical Gaussian clusters with diagonal covariance matrices of 0.5

Entities	features.	Clusters
1000	8	2
1000	12	3
1000	16	4
1000	20	5

3.9.2 Real-world Dataset

Thirteen different real-world datasets are obtained from the popular UCI machine learning repository. These datasets have both numerical and categorical features, as shown in the table below.

TABLE 3.2: Real-world datasets used in our experiments

			Original features		
DB Name	Entitites	Cluster	Numerical	Categorical	Total
Australian credit	690	2	6	8	14
Breast cancer	699	2	9	0	9
Car evaluation	1728	4	0	6	6
Ecoli	336	8	7	0	7
Glass	214	6	9	0	9
Heart	270	2	6	7	13
Ionosphere	351	2	33	0	33
Iris	150	3	4	0	4
Soybean	47	4	0	35	35
Teaching Assistant	151	3	1	4	5
Tic Tac Toe	958	2	0	9	9
Wine	178	3	13	0	13
Zoo	101	7	1	15	16

Note:unlike synthetic dataset, there is only one copy of real-world dataset.

Categorical Features to Binary Features

Each categorical feature v with M categories are replaced by M binary features where all the binary features are set to 0 except the one related to the binary feature is set to 1. For example, the A4 feature in the Australian credit dataset has three categories- p, g, and gg. The A4 feature is replaced by three binary features with 0 values initially. To represent the p, the first binary feature is set to 1 and the rest to 0. Likely, to represent gg value for A4, first two binary features are set to 0 and the last binary feature corresponding to gg is set to 1.

TABLE 3.3: Australian Credit Dataset

Feature	Data Values	No. Binary Features	Total Features
A1	a,b	2	2
A2	continuous	-	1
A3	continuous	-	1
A4	p,g,gg	3	3
A5	ff, d, i, k, j, aa, m, c, w, e, q, r, cc, x	14	14
A6	ff, dd, j, bb, v, n, o, h, z	9	9
A7	continuous	-	1
A8	t, f	2	2
A9	t, f	2	2
A10	continuous		1
A11	t, f	2	2
A12	s, g, p	3	3
A13	continuous	-	1
A14	continuous	-	1

Therefore, in total there are 2+1+1+3+14+9+1+2+2+1+2+3+1+1= 43 pre-processed features (excluding the class feature) in the Australian credit datasets.

3.9.3 Noise

To test the resistance of our algorithms against noise, we have added noise in the datasets. We have considered three factors to generate noise: noise strength, noise distribution, and noise model.

Noise Strength

The strength of noise measures the influence of the noise over the original data. The higher the strength of noise, the greater the chances of deviation of information from its originality. In this research we have tested noise strength at three levels:

- no noise
  The dataset is the same as the original dataset. Therefore, no synthetic noise is added in the original dataset.
- half noise
  In this category, we try to simulate the effect of noise by adding half the amount of noise relative to the feature's cardinality and the model of noise.
- full noise
  Here, we try to simulate the effect of noise by doubling the amount of noise relative to half noise.

The exact amount of noise in half and full noise varies on the distribution and model of noise we consider.

Noise Distribution

Gaussian and uniform noise are well studied in image processing [154, 155, 156, 157]. In our research, we have run our simulation in the dataset adding noise generated from Gaussian and uniform distribution.

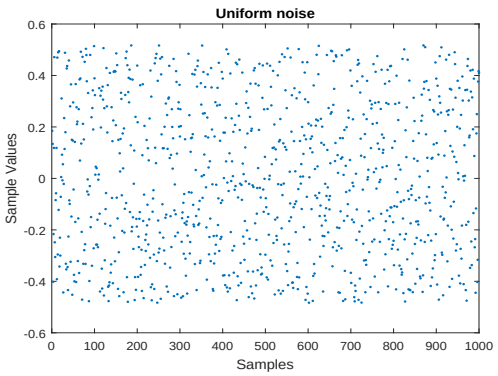


FIGURE 3.8: Uniform noise generated in the range [0,1].

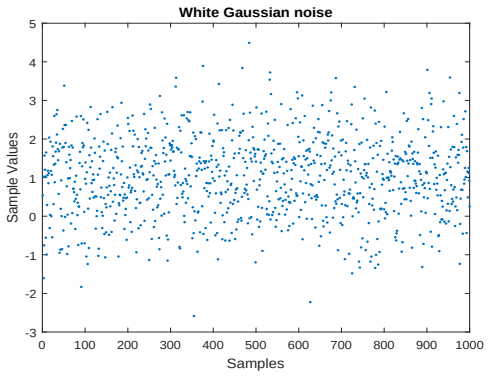


FIGURE 3.9: Noise generated from Gaussian distribution with mean = 0 and standard deviation = 1.

- Uniform Noise
  To generate uniform noise, we generate random values in the range (1,100). These values are the normalize before added to observe dataset.
- Gaussian Noise Gaussian noise is generated using a univariate Gaussian model with mean 0 and standard deviation 1. Again, similar to uniform noise, these noises are also standardized before adding to target dataset.

Noise Model

Noise present in a dataset can be broadly categorized as class or feature noise [158, 159]. Class noise refers to the deviation of a class label from its original value. In supervised learning, class noise present in a training dataset has greater influence over the performance of the classification algorithm. However, they have no role in unsupervised learning unless semi-supervised learning is considered. Therefore, in our experiments, we just deal with the feature noise. There are three type of feature noise considered in this thesis: added noisy features, feature blure and cluster based feature blure.

- Noisy Feature Added (NFA): extra noisy features are added in a dataset.
- Feature Blurring (FB): selected number of features are blurred by adding noise.
- Cluster-based Feature Blurring (CFB): selected number of features are blurred by adding noise but only within a particular cluster.

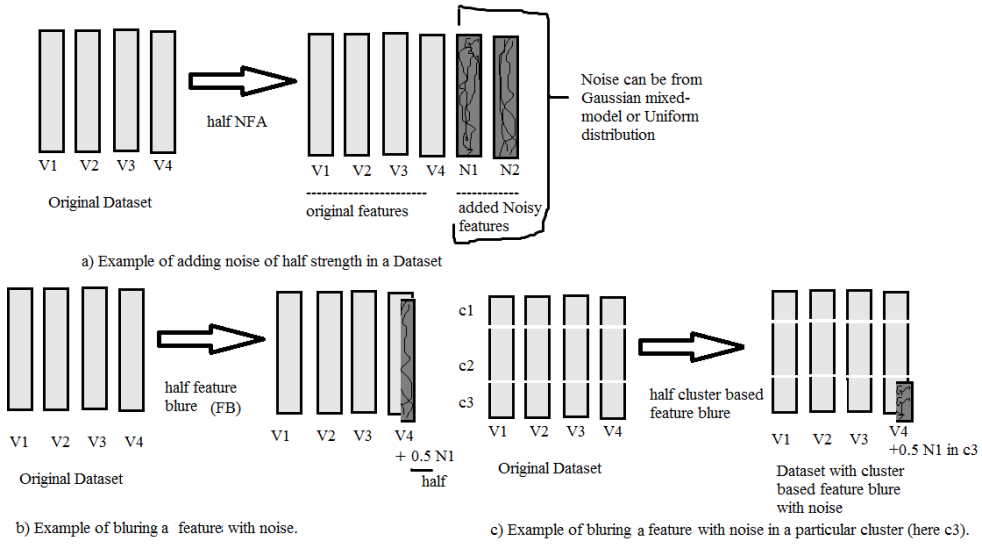


FIGURE 3.10: Examples of adding noise in dataset.

In sub-figure, Figure 3.10.a, there are four features V_1, V_2, V_3 and V_4 in the original dataset. For half strength of added noise, two noisy features N_1 and N_2 are added in the dataset resulting total of six features. In sub-figure Figure 3.10.b, V_4 is selected (randomly) and blurred with 50% of noise N_1 for half strength feature blur noise. In sub-figure Figure 3.10.c, cluster c_3 is chosen (randomly) and its feature V_4 is blurred with 50% of noise N_1.

3.9.4 Noise Configuration

To add noise in a dataset, at first one of the noise distributions: Uniform or Gaussian is chosen. Then noise strength is fixed- half or full, and finally one of the noise models- noisy feature added (NFA), feature blurring (FB), or cluster-based feature blurring (CFB) is selected. Therefore, for a particular dataset, there will be 13 copies of the datasets- 12 with added noise and one without any noise.

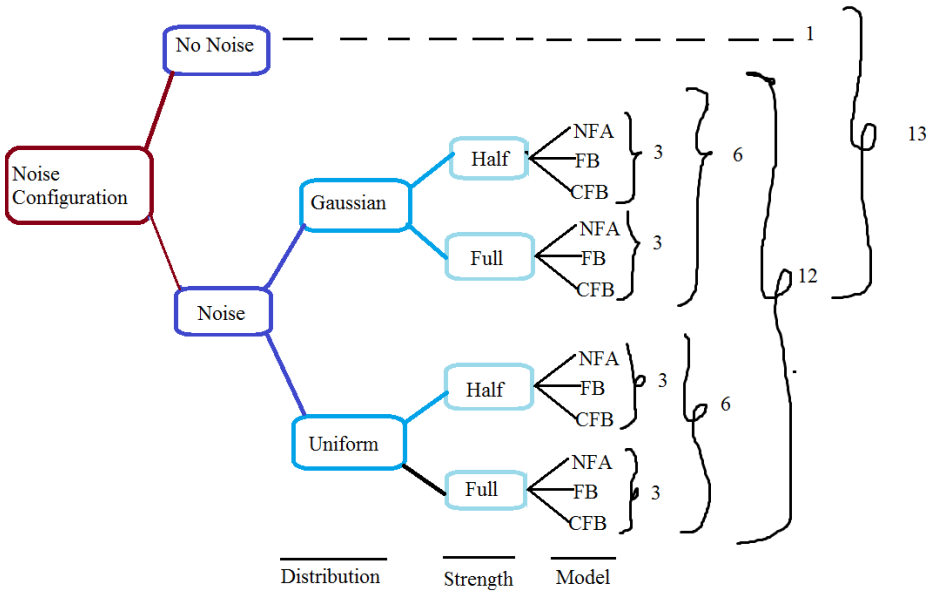


FIGURE 3.11: Noise configuration: there are two noise distributions, two noise strength and three noise models.

In a dataset Y is defined in a V feature space which have $S = S_1, S_2, ..., S_k$ clusters. The 13 datasets given by configuration in Figure 3.11 are:

- d_0 No Noise: is the original dataset Y with feature space cardinality $|V|$.
- d_1 Gaussian-half-NAF: there will be $\frac{|V|}{2}$ extra noisy features were generated from the Gaussian mixed-model will be added in the dataset.
- d_1 Gaussian-half-FB: random $\frac{|V|}{2}$ features are chosen and are be blur with noise generated from the Gaussian mixed-model.
- d_2 Gaussian-half-CFB: random $\frac{|V|}{2}$ features from one of the random cluster S_k are blur with noisy generated from the Gaussian mixed-model.
- d_4 Gaussian-full-NAF: there will be $|V|$ extra noisy features generated from the Gaussian mixed-model will be added in the dataset.
- d_5 Gaussian-full-FB: random $|V|$ features are chosen and are be blur with noise generated from the Gaussian-mixed model.
- d_6 Gaussian-full-CFB: random $|V|$ features from one the random cluster S_k are blur with noisy generated from the Gaussian-mixed
- d_7 Uniform-half-NAF: there will be $\frac{|V|}{2}$ extra noisy features generated from the Uniform distribution will be added in the dataset.
- d_8 Uniform-half-FB: random $\frac{|V|}{2}$ features are chosen and are be blur with noise generated from the Uniform distribution.
- d_9 Uniform-half-CFB: random $\frac{|V|}{2}$ features from one of the random cluster S_k are blur with noisy generated from the Uniform distribution.
- d_{10} Uniform-full-NAF: there will be $|V|$ extra noisy features generated from the Uniform distribution will be added in the dataset.

- d_{11} Uniform-full-FB: random $|V|$ features are chosen and are be blur with noise generated from the Uniform distribution.
- d_{12} Uniform-full-CFB: random $|V|$ features from one of the random cluster S_k are blur with noisy generated from the Uniform distribution.

3.9.5 Datasets and Noise Configurations for Experiments

Different datasets used in this thesis are:

- Chapter 5 :- [Feature Selection]
  Synthetic Datasets - four GMM configurations (as listed in Table 3.1) X 20 copies
  UCI datasets- seven UCI datasets (Echoli, Glass, Heart, Iris, Soya, Wine, Zoo)
  Noise - thirteen noise configurations (from Figure 3.11).
- Chapter 6 :- [Missing Values]
  Synthetic Datasets - four GMM configurations (as listed in Table 3.1) X 20 copies
  UCI datasets - seven UCI datasets (Echoli, Glass, Heart, Iris, Soya, Wine, Zoo)
  Noise - thirteen noise configurations as listed in Figure 3.11
  Percentage Missing - seven missing percentages (1%,10%,15%,20%,25%,30%,35%)
  Noise - three Noise configurations (no noise, Half NFA, Full NFA)
  Missing Pattern - two missing pattern (Missing Completely at Random and Not Missing at Random)
- Chapter 7:- [p-norms]
  - Validation of proposed Algorithm
    Dataset- a GMM cluster in 2D with center (5,10).
  - Transformation of Data in different p-norms
    Datasets- three GMM clusters with five features
    p-norms- copies of each dataset in p-norms [p range from 1.1 to 5, in interval of 0.1]
    Noise - three Noise configurations (no noise, Half NFA, Full NFA, from Figure 3.11)
- Chapter 8 :- [Extended Experiments]
  Synthetic Datasets - four GMM configurations (as listed in Table 3.1) X 20 copies
  UCI datasets- thirteen UCI datasets (as listed in Table 3.2)
  Noise - three Noise configurations (no noise, Half NFA, Full NFA from Figure 3.11)
  p-norms - copies of each dataset in p-norms [p range from 1.1 to 5, in interval of 0.1]

Chapter 4

Proposed Algorithms

This chapter contains the important algorithms developed within this thesis. The algorithms are: the feature-selection algorithm discussed in Chapter 5, the partial-distance measured used in Chapter 6, and the algorithm to transform data between different p-norms proposed in Chapter 7.

4.1 Feature selection

4.1.1 Feature selection via Feature Weighting

The proposed approach, Feature Selection via Feature Weighting (FSFW), is built on previous work in feature weighting by using the *iMWk*-Means as a tool to find feature weights. These feature weights are then used as an index for feature selection. The cluster based feature weighting, w_{kv}, discussed earlier in equation 3.9 models the degree of importance of feature v at a cluster S_k. The proposed method first maps the cluster based feature weighting $w_{kv} \in [0, 1]$ to $w_v \in 0, 1$ in order to select or deselect a feature v. The key to this mapping is to find a threshold θ so that

$$w_v = \begin{cases} 1, \text{ if } w_v \geq \theta, \\ 0, \text{ if } w_v < \theta. \end{cases} \tag{4.1}$$

Two different feature selection methods, *mean*-FSFW and *max*-FSFW, are proposed in this thesis based on the approach discussed above. The cluster based feature weighting obtained from *iMWk*-Means has K weights : $w_{kv}, k = 1, 2, ..., K$, although each feature v has one applicable weight (4.1). In the first method, *mean*-FSFW, w_v is set to $K^{-1} \sum_{k=1}^{K} w_{kv}$ whereas in the second method, *max*-FSFW, w_v is set to $max(\{w_{1v}, w_{2v}, ..., w_{Kv}\}$. In both algorithms, θ is set to m^{-1} and w_v is set to 0 or 1 using (4.1). Features with $w_v = 0$ are then removed from the data set.

Selection of θ
Let us assume for a dataset, Y, all features are drawn from a truly random and uniform distribution. This means all features have uniform distribution, i.e. $D_{kv} \equiv c$, where c is a constant. The feature weights are normalized to sum to 1. Therefore each feature weight $w_{kv} \equiv \frac{1}{m}$.

Note:
The code for the proposed *mean*-FSFW and *max*-FSFW algorithms is available in the repository at `https://bitbucket.org/m-learning/phd/src/master/`

4.2 Partial Distance for Handling Missing Values

4.2.1 wk-MeansPD

Weighted K-Means with partial distance(WK-MeansPD)

Weighted *k*-Means with partial distance, *Wk*-MeansPD, uses partial distance to calculate similarity between data entities with missing attribute values. Similar to *k*-MeansPD, *Wk*-MeansPD does not require additional imputation of values and follows the same basic steps as *Wk*-Means, with some modifications to address missing values.

An initial K centroids are chosen from a clean dataset in order to avoid missing values in centroids. Centroids in each iteration are updated to the mean of the cluster over known attribute values. Weights for each iteration are changed based on the dispersions calculated over the available attribute values. Then, the weighted distance function 3.15, used to calculate the similarity between entities, is replaced with the weighted partial distance.

$$PD_w(S_{y_i,y)_j}) = \frac{oF + mF}{oF} \sqrt{w_{kv} \sum_{j=1}^{N} (Dist_j)^2} \tag{4.2}$$

where, w_{kv} is the weight of the attribute calculated using the concept of partial distance from equation (3.9).

4.2.2 Minkowski Weighted *k*-Means with partial distance(*MWk*-Means*PD*)

The Minkowski weighted *k*-Means with partial distance, *MWk*-Means*PD*, an extension of *Wk*-Means*PD* to Minkowski space, uses partial distance in L_p norm to calculate the similarity between two entities with missing attribute values. The "Minkowski centres" [153] that minimise the sum of the distance of each entity within a cluster to its centroids in L_p norm is calculated using the partial distance. Other steps remain similar to *Wk*-Means*PD*, as defined in section 4.2.1.

Note:
The code for the proposed *Wk*-Means*PD*, *iWk*-Means*PD*, *MWk*-Means*PD* and *iMWk*-Means*PD* are available in the repository at
`https://bitbucket.org/m-learning/phd/src/master/`

4.3 Transformation of Dataset to Different p-norms

By default, the Gaussian mixed model, gmm in MATLAB, uses 2-norm (Euclidean distance) to produce clusters in the Cartesian coordinate system. In this thesis, a p-norm data transformation procedure (*pNorm*TransInCartesian), is designed to transform a dataset, which is represented by Cartesian coordinates, from p_{old}-norm to p_{new}-norm. The transformation procedure has three basic steps.

Algorithm 1 *pNorm*TransInCartesian

1. Move the coordinates representation of data points from Cartesian to Polar, see appendix A.1.
2. Transform the data points from p_{old}-norm to p_{new}-norm, using algorithm 2 below.
3. Move the coordinates back to Cartesian system.

The process of transforming data points from 2-norm to p-norms in Cartesian coordinates, validation of the procedure and application of the procedure in GMM clusters, are explained in detail below.

4.3.1 Transformation of a Dataset from p_{old}-norm to p_{new}-norm

Transformation of a dataset from p_{old}-norm to p_{new}-norm in polar coordinates can simply be done by moving each data point relative to the pole by a ratio of the radii, $\frac{r_{new}}{r_{old}}$, towards polar axes where, r_{old} and r_{new} are the radius of the dataset in p_{old}-norm and p_{new}-norm respectively. To carry out experiments with synthetic datasets in different p-norms, Gaussian clusters are first created in 2-norm using the MATLAB built in Gaussian mixed model. These datasets are then translated to respective p-norms using six steps. This algorithm is explained and validated in Chapter 7, section 7.2.

Algorithm 2 p_{old}-normTop_{new}-normInPolar

1. Relocate data center to origin.
2. Relocate all data points to positive plane*.
3. Shrink points towards or expand from the origin.
4. Relocate all data points back to their respective plane.
5. Relocate center of the data points back to its original location.

Note*:
The term positive plane refers to the location where all axes of the coordinates are positive. E.g. first quadrant in 2D.

Transformation Steps:

a. **Relocate data center to origin** [see Figure 7.1.b]
Relocation of the center of the cluster to the origin of the coordinate system is obtained by displacing all data points towards the origin of the coordinates. This displacement is equal to the vector between the center of the data points and the origin. Mathematically, we can summarize this step as:

   i. center = Minkowski centroid [153] of the data points.
   ii. displaced data points = data points - center.

b. **Relocate all data points to positive plane** [see Figure 7.1.c]

This can be simply achieved by considering only the absolute deviation of a data point from its axis. The sign of the deviation, SIGN, is saved as it is required to reverse the process in step d.
SIGN = -1*cluster ≤ 0 + cluster ≥ 0 . cluster in positive plane = abs(displaced cluster).

c. **Shrink toward or expand from the origin** [see Figure 7.1.d]
Each data point is shrunk toward or expanded from the origin by the ratio of the distance of the data point from its origin in the two p-norms.

data points in positive plane in p_{new}-norm = $\frac{r_{new}}{r_{old}}$ X data points in positive plane.

d. **Relocate all data points back to their respective plane** [see Figure 7.1.e]

To relocate each data point to its original plane, the relocated data points are simply multiplied by the SIGN value which is obtained in step b.

data points back to respective quadrant = (data points in positive plane in p_{new}-norm) * SIGN.

e. **Relocate center of the data points back to their original location**[see Figure 7.1.f] data points in p_{new}-norm = data points back to respective plane + center

Note:
How SIGN is calculated:-

Let cluster S_k= $\begin{bmatrix} 4 & 3 & -5 \\ -8 & 9 & 6 \\ 2 & 4 & -7 \end{bmatrix}$

SIGN = -1*$S_k \leq 0$ + cluster ≥ 0

$$\Rightarrow \text{-1*} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

Moving cluster to positive plane:

S_k in positive plane = $|S_k|$

$$\Rightarrow \left| \begin{bmatrix} 4 & 3 & -5 \\ -8 & 9 & 6 \\ 2 & 4 & -7 \end{bmatrix} \right|$$

$$\Rightarrow \begin{bmatrix} 4 & 3 & 5 \\ 8 & 9 & 6 \\ 2 & 4 & 7 \end{bmatrix}$$

Finally, the data points in p_{new} space are recovered by moving the center of the reshaped data points from the origin of the coordinates to their original center. This can be achieved by reversing the process which is previously carried out in step a.

Chapter 5

Feature Selection

Feature selection, one of the major pre-processing techniques in machine learning, helps in optimizing feature space cardinality by selecting the most informative features. Reduction of features space by removing redundant and noisy features will help to reduce execution time, lower storage requirements and fix, curse of dimensionality, problems in data mining algorithms. However, maintaining the important information within the reduced feature space is a challenging task. In this research, feature weighting is used as a technique to choose the most representative features for a dataset. The weighted variants of *k*-Means, derives feature weights per clusters based on their distribution. Feature weights per cluster can be combined in multiple ways to derive a common weight for a particular feature over the whole dataset.

In this set of experiments, weights are combined in two ways, i.e. mean feature weight per cluster and maximum feature weight per cluster. The first method is called feature selection based on mean feature weight within clusters (*mean*-FSFW) and the second one, feature selection based on maximum feature weight within clusters(*max*-FSFW). These two novel feature selection algorithms are derived and tested on both real and synthetic datasets. Experimental results from these feature selection algorithms are presented and analysed. Aims to be achieved through these experiments are listed in section 5.1. The experimental set up is discussed in section 5.1. Results from the experiments are presented in section 5.3 and finally, conclusions from the results are presented in section 5.4

5.1 Aims

Aims of this experimental sets are:

- **To conduct the proposed two feature selection algorithms against other algorithms**

  Performance of these two weighted based feature selection algorithms is observed against three existing feature selection algorithms. These three feature selection algorithms are feature selection based on feature similarity (FSFS), multi-cluster feature selection (MCFS) and intelligent *k*-Means feature selection. FSFS and MCFS are the most popular feature selection algorithms. IKFS uses a feature weighting principle for feature selection which is similar to our proposed algorithms. These algorithms are compared with *k*-Means and intelligent Minkowski weighted *k*-Means (*iMWk*-Means) as a benchmark.

- **To conduct the comparison in both synthetic and real-world datasets.**

  Experiments are run on both synthetic and real-world(UCI) datasets. Results from the synthetic datasets show how the proposed algorithms perform under a known configuration whereas results from real-world datasets help to predict how they behave in a real-world scenarios.

- **To observe the impact of noise on the performance of algorithms.**

  Three different types of noise configuration are chosen to observe the performance of these feature selection algorithms. These noise configurations include two types of noise distributions such as Gaussian and Uniform, three types of noise model, i.e. adding noisy features, blurring features with noise and blurring features from a selected cluster with noise.

- **To extend our study on Minkowski space**

  The proposed algorithm is extended in the Minkowski space where the distance coefficient (p) varies from 1.1 to 5.0, with interval of 0.1. The main purpose of this extension is to examine the impact of distance coefficient (p) in feature selection.

5.2 Experimental Set-up

5.2.1 Dataset Set-up

Experiments are run in both synthetic and real-world (UCI) datasets. As mentioned in Chapter 3, Section 3.9, there are thirteen different configurations for the real-world datasets and 4 different configurations for synthetic datasets. The experiments are run twenty times in each configuration, averaging the results to reduce the impact of random variations, such as the position of the centroids.

Impacts of noise on the performance of these algorithms are observed by adding noise to the dataset. As described in Chapter 3, Section 3.9, three levels of noise strength are considered: no noise, half noise and full noise; two types of noise distribution: Gaussian and Uniform; and three types of noise model: adding extra noisy features (NFA), blurring features by noise (FB) and blurring features by noise within a cluster (CFB). Therefore, for each dataset there are two (noise strength- as no noise is not considered here) times two (noise distribution) times three (noise model) equals twelve plus one(no noise) replicates of dataset with and without added noise.

There are four different synthetic dataset configurations with twenty copies of dataset in each configuration. This gives four (data configurations) times twenty(copies of datasets) times thirteen (noise configuration) which equals to 1040 sets of the synthetic dataset. In the real-world (UCI) datasets there are thirteen datasets times twelve plus one (noise configuration), which is 169 set of real-world dataset for the experiment.

5.2.2 Parameter Tuning

Four feature selection algorithms which are used in these experiments required parameter tuning. FSFS and MCFS required the number of features to be selected in advance. In our proposed algorithms the value of the the distance coefficient (p) needs to be known in advance. The Silhouette index used to tune these parameters. The number of nearest neighbours is another parameter required to tune MCFS. The number of nearest neighbours is set to five as a default value, suggested by author[24]. Under the supervised condition (data labels given), the best combination of selected features in MCFS, FSFS, *mean*-FSFW and *max*-FSFW can be found by matching the data labels obtained from these algorithms. These experiments are named “Best Case". Achieving the “Best Case" in FSFS, MCFS, *mean*-FSFW and *max*-FSFW is practically impossible under unsupervised learning. However, “Best Case" can be used as a benchmark.

Both proposed algorithms use anomalous clustering to find initial centroids. A similar approach has been used in intelligent *k*-Means feature selection, IKFS. So, it worthwhile to compare the proposed algorithms against the IKFS as well.

Performance of the proposed algorithms is also compared against *k*-Means and intelligent weighted *k*-Means (*iWk*-Means). These two variants of *k*-Means are used to evaluate whether feature selection makes any improvement in cluster recovery, especially with added noise.

5.3 Result Presentation

Two factors to be considered while displaying the results:

- Organization of Results
- Heading of Table

Two sections below describe how results are organized and how to read the data table.

5.3.1 Organization of Results

Results from the experiments are organised as follow:

1. Results from Synthetic datasets
2. Results from real-world datasets

Simulations on the synthetic datasets with and without noise are executed first. This will help in observing how the proposed algorithms work under known distributions. Later, simulations are repeated in the UCI datasets.

5.3.2 Results from Synthetic Datasets

Results from the synthetic datasets are further summarized based on three noise configurations:

1. Noise strength
2. Noise distribution
3. Noise model

The results obtained from the synthetic datasets are summarized under three subheadings: noise strength, noise distribution and noise model.

5.3.3 Organizing from Real-World Datasets

Results from the real-world (UCI) datasets are analysed and presented as in the approach used earlier with the synthetic datasets. Results from the UCI datasets are summarized base on:

1. Noise strength
2. Noise distribution
3. Noise model

5.3.4 Table Organization

Columns in each table are divided into two subheadings: configuration and performance measurement. The first subheading, configurations covers one or two columns that contain information about the configuration of noise of the dataset. The second subheading, performance measurement, contains results from nine different algorithms under the selected configuration.

Each table is divided horizontally, into two partitions based on the metric used for the performance. Cluster recovery index (ARI Index) is used in the first horizontal partition, whereas the second horizontal partition measures the performance based on the percentage of feature space cardinality after feature selection. The best results in the corresponding observations(row) are indicated in the bold font.

Twenty different datasets are generated for each configuration in synthetic datasets. Each value in the performance measurement denotes the mean of these readings and the mean of standard deviation of the 20 datasets, and are separated by "/". Only a single dataset is available in the real-world (UCI) dataset for each configuration. Therefore, only the mean of the performance measurements is used.

The first two columns under the cluster recovery subheading contain results from *k*-Means and *iWk*-Means. Since there is no feature selection in these two algorithms, their feature selection reads 100 percent. The third column under the subheading results from the intelligent *k*-Means feature selection. The remaining four columns contain results from FSFS and MCFS, with records from silhouette parameter tuning and the best case (supervised) for each of them. The last two columns show results from our proposed algorithm "feature selection using feature weighting (FSFW)"- for *mean*-FSFW and *max*-FSFW respectively. The *mean*-FSFW uses mean feature weights within clusters as the selection criteria whereas *max*-FSFW uses the maximum feature weight along clusters as the feature selection criteria. Headings/sub-headings of tables are summarised as:

1. Configuration
2. Performance Measurement
   (a) benchmark
      i. *k*-Means
      ii. *iWk*-Means
   (b) iKFS
   (c) FSFS
      i. tuning parameter using silhouette index
      ii. best case (supervised)
   (d) MCFS
      i. tuning parameter using silhouette index
      ii. best case (supervised)
   (e) FSFW
      i. *mean*-FSFW
      ii. *max*-FSFW

5.3.5 Synthetic Datasets

Four different dataset configurations with 20 datasets per configuration are used in this experiment. These datasets are generated from the mixed model Gaussian distribution (details are explained in Chapter 3, Section 3.9. The three tables below summarize the results from the synthetic datasets. In the first table, different feature selection algorithms are compared under the unsupervised condition where the Silhouette index is used for parameter tuning. In the second table, the best case for each algorithm are compared, whereas in the third table, the effects of the distance coefficient on the performance of the proposed algorithms are presented.

- **Parameter Tuning with Silhouette Index**

  In table 5.1, the performances of four different feature selection algorithms are compared with two variants of *k*-Means (no feature selection) using the ARI index and % of feature selected.

  TABLE 5.1: Comparision of four different feature selection algorithms with original var of *k*-Means and intelligent Minkowski weighted *k*-Means in synthetic datasets with and without added noise. Silhoutee index is used for parameter tuning.

		Performance measurement [mean/std] ARI						
		Without FS		With FS				
							FSFW	
		k-Means	*iMWk*-Means	iKFS	FSFS	MCFS	*mean*-FSFW	*max*-FSFW
avg cluster Recovery Index	Noise Strength							
	No	**0.974**/0.02	**0.974**/0.03	**0.974**/0.02	0.183/0.22	0.277/0.24	0.68/0.22	0.915/0.09
	Half	0.488/0.07	**0.942**/0.07	0.485/0.07	0.138/0.16	0.34/0.2	0.685/0.25	0.862/0.15
	Full	0.408/0.05	**0.867**/0.12	0.407/0.05	0.139/0.15	0.322/0.2	0.729/0.23	0.81/0.17
	Noise Distribution							
	Gaussian	0.488/0.04	**0.919**/0.08	0.487/0.04	0.144/0.16	0.319/0.2	0.685/0.25	0.835/0.16
	Uniform	0.407/0.08	**0.89**/0.11	0.405/0.08	0.133/0.16	0.344/0.2	0.729/0.23	0.837/0.17
	Noise Model							
	NFA	0.906/0.08	0.912/0.11	0.906/0.09	0.126/0.2	0.414/0.16	0.821/0.18	**0.958**/0.05
	FB	0.014/0.01	0.839/0.12	0.013/0.01	0.137/0.17	0.326/0.29	0.629/0.28	**0.902**/0.14
	CFB	0.424/0.09	**0.963**/0.05	0.42/0.08	0.154/0.1	0.253/0.16	0.672/0.26	0.648/0.29
avg % of feature selected	Noise Strength							
	No	-	-	96.9/4.6	9.9/4.3	**9.3**/3.9	26.4/14.8	47.5/13.6
	Half	-	-	97.3/4.1	10/4.7	**8.3**/2.6	29.9/16.1	50.8/13.1
	Full	-	-	94.7/6.3	14.5/9.3	**9.1**/4.8	32.9/15	54.8/11.8
	Noise Distribution							
	Gaussian	-	-	96.1/5	12.6/7.5	**8.6**/3.9	29.4/15.4	51.8/13
	Uniform	-	-	95.9/5.4	12/6.5	**8.8**/3.5	33.4/15.7	53.8/12
	Noise Model							
	NFA	-	-	98.9/1.7	5.4/2	**4.8**/0.2	31.8/13.1	44.6/8
	FB	-	-	91.7/9.7	**9.1**/3.1	10/4.5	32.7/17.8	60.1/12.5
	CFB	-	-	97.3/4.1	22.3/15.8	**11.4**/6.3	29.7/15.7	53.7/17

  - **Noise Strength**

    In table 5.1, *k*-Means, *iWk*-Means and IKFS provide better cluster recovery rate with ARI Index 0.974 and the deviation between the readings is also least (around 0.03) when there is no noise in the dataset. However, these observations are obtained under no feature selection in *k*-Means and *iWk*-Means whereas IKFS has no contribution towards the feature selection. IKFS has 96.9% of average features selection. The closest best performance with no noise in the dataset is observed with *max*-FSFW (ARI Index 0.915) followed by *mean*-FSFW (ARI Index 0.68), MCFS (ARI Index 0.277) and

FSFS (ARI Index 0.183). However, MCFS has the lowest percentage of feature selection (9.3%), followed by FSFS (9.9%), *mean*-FSFW (26.4%) and *max*-FSFW (47.5%).

In the case of half noise, *iWk*-Means performance best (ARI Index 0.942), followed by *max*-FSFW (ARI Index 0.862), *mean*-FSFW (ARI Index 0.685), IKFS (ARI Index 0.485), MCFS (ARI Index 0.34) and FSFS (ARI Index 0.485). In terms of the percentage of feature selection, MCFS performance best (8.3%) followed by FSFS (10%), *mean*-FSFW (29.9%), *max*-FSFW (50.8%) and iKFS (97.3%).

In the case of full noise, *iWk*-Means provides best cluster recovery (ARI Index 8.67), followed by *max*-FSFW (ARI Index 0.81), *mean*-FSFW (ARI Index 0.729), IKFS (ARI Index 0.407), MCFS (ARI Index 0.322) and FSFS (ARI Index 0.139). Furthermore, observation with percentage of feature selection shows that MCFS gives the best result (9.1%) followed closely by FSFS (14.5%), *mean*-FSFW (32.9%), *max*-FSFW (54.8%) and iKFS (94.7%).

Overall, *iWk*-Means offers better resistance to noise than *k*-Means. Addition of noise has decreased the performance of all algorithms with the exception in *mean*-FSFW where addition of noise has consistently increased the ARI index from 0.68, 0.685 and 0.729 for no noise, half noise and full noise respectively. However, deviation of the results (mean standard deviation) of *mean*-FSFW is relatively higher and is constantly increasing with noise. It is also observed that MCFS provides the best feature reduction with almost 90% of dimension reduction is observed, but its cluster recovery (ARI Index) is comparatively low. FSFS also gives better dimension reduction but cluster recovery (ARI Index below 0.2). Hence it can not be considered as a choice for noisy data while IKFS shows no sign of feature space reduction as it stores at least 94% of features selection. Comparatively, *max*-FSFW gives the best cluster recovery rate with at least 0.81 ARI Index and its performance consistently increases with added noise as the deviation between the *iWk*-Means diminishes with increase in noise strength.

– **Noise Distribution**

As stated earlier, performance of algorithms is categorized based on two noise distribution, i.e. Gaussian and Uniform. *max*-FSFW gives better cluster recovery (ARI Index 0.835) among the feature selection algorithms followed by *mean*-FSFW (ARI Index 0.685), IKFS (ARI Index 0.487) when noise is generated from Gaussian distributions. However, MCFS has better dimension reduction (only 8.6% features selected), closely followed by FSFS (12.6%), *mean*-FSFW (26.4%), *max*-FSFW (51.8%) and IKFS (96.1%). A similar pattern of performance, *max*-FSFW followed by *mean*-FSFW, iKFS, MCFS and FSFS, is observed when noise is generated from Uniform distributions.

FSFS and MCFS provide a high percentage of feature space reduction but low cluster recovery in both cases. *mean*-FSFW and *max*-FSFW both surpass *k*-Means but cannot improve over the weighted variant of *k*-Means (*iWk*-Means here) in term of cluster recovery (ARI Index). *max*-FSFW outperforms *mean*-FSFW based on cluster recovery whereas *mean*-FSFW is able to reduce almost two-thirds of feature space cardinality. *k*-Means, *iWk*-Means, IKFS and FSFS give better cluster recovery when noise is from Gaussian distributions. Similarly, *mean*-FSFW,maxFSFS and MCFS provide better cluster recovery when datasets have noise from uniform distributions.

– **Noise Model**

max-FSFW provides better cluster recovery (ARI Index 0.958) than non feature selection based algorithms *iWk*-Means (AIR Index 0.912) and *k*-Means (ARI Index 0.90, and feature selection based algorithms IKFS (ARI Index 0.906), *mean*-FSFW (ARI Index 0.821), MCFS (ARI Index 0.414) and FSFS (ARI Index 0.126) with the "added noisy feature" noise model. Best feature selection is observed in MCFS (4.8% of total feature selected), followed by FSFS (5.4% feature selection), *mean*-FSFW (31.8 % feature selection), *max*-FSFW (44.6% features selected) and iKFS (98.9% features selected). However, MCFS and FSFS provide better feature space reduction and their cluster recovery is relatively low. Unlike other configurations, with noisy features added to the dataset, IKFS has no contribution to offer feature selection.

Blurring features with noise significantly reduces cluster recovery in *k*-Means and IKFS (ARI Index 0.014 and 0.013 respectively) whereas *max*-FSFW gives the highest cluster recovery (ARI Index 0.902), followed by *iWk*-Means (ARI Index 0.839), MCFS (ARI Index 0.326) and FSFS (ARI Index 0.137). As in earlier cases, MCFS gives best feature space reduction, followed by FSFS, *mean*-FSFW and *max*-FSFW.

Overall, *max*-FSFW provides best cluster recovery with more than 55% and 40% feature space reduction when datasets have noisy feature added or feature blurring whereas MCFS provides best feature reduction rate but is less effective as average cluster recovery is around 30%.

- **Best Case (Supervised Learning)**

In table 5.1 above, Silhouette index has been used for tuning parameters in FSFS, MCFS, *mean*-FSFW and *max*-FSFW. In table 5.2 below, the best possible results from each feature selection algorithm are compared with *k*-Means and intelligent Minkowski weighted *k*-Means, *iMWk*-Means. As mentioned earlier, best possible cases are obtained under the assumption that labels are known in advance although it is impossible to achieve the best cases in the unsupervised environment (through parameter tuning). However, these figures give an insight into the best performance which can be achieved theoretically by the particular algorithm .

TABLE 5.2: Comparison of four different feature selection algorithms with original variate of KMeans and intelligent Minkowski Weighted KMeans in synthetic dataset under the supervised condition (best possible cases).

		Performance Measurement [mean/std]						
					Best Possible Cases			
		kMean	iWKMeans	iKFS	FSFS	MCFS	meanFSFW	maxFSFW
avg cluster Recovery Index	Noise Strength							
	No	**0.974**/0.02	**0.974**/0.03	**0.974**/0.02	0.959/0.03	0.971/0.03	0.954/0.04	0.97/0.03
	Half	0.488/0.07	0.942/0.07	0.485/0.07	0.83/0.17	0.746/0.2	**0.969**/0.03	0.935/0.08
	Full	0.408/0.05	0.867/0.12	0.407/0.05	0.819/0.19	0.755/0.21	**0.969**/0.03	0.939/0.08
	Noise Distribution							
	Gaussian	0.488/0.04	0.919/0.08	0.487/0.04	0.838/0.17	0.813/0.17	**0.968**/0.03	0.958/0.05
	Uniform	0.407/0.08	0.89/0.11	0.405/0.08	0.811/0.19	0.688/0.23	**0.97**/0.03	0.915/0.11
	Noise Model							
	NFA	0.906/0.08	0.912/0.11	0.906/0.09	0.957/0.04	0.911/0.08	**0.975**/0.03	**0.975**/0.03
	FB	0.014/0.01	0.839/0.12	0.013/0.01	0.724/0.28	0.62/0.33	**0.966**/0.03	**0.966**/0.03
	CFB	0.424/0.09	0.963/0.05	0.42/0.08	0.791/0.2	0.72/0.2	**0.966**/0.03	0.869/0.17
avg % of feature selected	Noise Strength							
	No	-	-	96.9/4.6	89/6.4	75.9/11.1	**50.1**/8.6	62.5/10.5
	Half	-	-	97.3/4.1	68/19.4	**53.1**/24.7	64.7/11.5	67/14.1
	Full	-	-	94.7/6.3	64.4/21.9	**46.5**/21.8	62.4/10.9	66.8/12.6
	Noise Distribution							
	Gaussian	-	-	96.1/5	70.2/19.5	**50.5**/21.2	62.2/11.4	67.9/13.2
	Uniform	-	-	95.9/5.4	62.3/21.8	**49**/25.3	64.9/11	65.9/13.4
	Noise Model							
	NFA	-	-	98.9/1.7	81.7/13.6	64.8/19.6	**47.5**/6.5	53.3/9.1
	FB	-	-	91.7/9.7	51.4/26.3	**31.8**/24	72.6/12.9	74.2/11.5
	CFB	-	-	97.3/4.1	65.6/22.2	**52.7**/26.2	70.5/14.2	73.1/19.4

– **Noise Strength**

k-Means, *iWk*-Means and IKFS provide better cluster recovery (ARI Index 0.974) than the best possible cases in four of the feature selection algorithms with no noise in the dataset. In this configuration, *mean*-FSFW gives the highest feature selection (around 50%), but lowest cluster recovery (ARI Index 0.954). With the addition of half noise, *mean*-FSFW has better cluster recovery, with 34% reduction on feature space. *mean*-FSFW and *max*-FSFW has better cluster recovery (best case) than *k*-Means and *iWk*-Means in full noise. In both cases, MCFS gives best feature selection (53.1% and 46.5%) in half and full noise; however it provides the lowest cluster recovery.

– **Noise Distribution**

mean-FSFW provides best cluster recovery (ARI Index 0.968), followed by *mean*-FSFW (ARI Index 0.958) when noise in the datasets are from a Gaussian distribution,. In this scenario, both *mean*-FSFW and *max*-FSFW are able to reduce feature space cardinality by 30% at least. Best feature space reduction is obtained by MCFS but with the lowest cluster recovery (ARI Index 0.813). *mean*-FSFW provides an optimal ARI index (0.97), followed by *max*-FSFW and *iWk*-Means with Uniform distribution. In the case of noise generated from Gaussian distribution, higher feature space cardinality is observed in MCFS but with the lowest cluster recovery (ARI Index 0.688).

Noise from Gaussian and Uniform distributions have similar role in the performance of feature selection algorithms. In general, all algorithms provide better cluster recovery against Gaussian noise but low feature space reduction, except *mean*-FSFW.

– **Noise Model**

mean-FSFW shows better cluster recovery(ARI Index 0.975) and a high percentage of feature space reduction (feature selection 47.5%) when noisy features are added to the dataset (NFA model). *max*-FSFW also provides similar cluster recovery as *mean*-FSFW but with higher feature selection (53.5%). These two algorithms are closely followed by FSFS and MCFS, and all four feature selection algorithms have higher cluster recovery than *k*-Means and *iMWk*-Means.

In the case of feature blurring noise, *mean*-FSFW and *max*-FSFW share the highest cluster recovery (ARI Index 0.966) followed by *iWk*-Means (ARI Index 0.839), FSFS (ARI Index 0.724) and MCFS (ARI Index 0.62); however, best feature space reduction is observed in MCFS (31.8%).

mean-FSFW gives best cluster recovery (ARI Index 0.966) with cluster based feature blur; however, percentage of original feature selection is relatively higher (70.5%) than MCFS (52.7%) and FSFS 65.6%.

- **Distance coefficient (p)**

In table 5.3 below, performance of *mean*-FSFW and *max*-FSFW based on three different values of distance coefficient (p) is examined. In the first and second case, p selected from a range of 1.1 to 5 with interval of 0.1 which maximized cluster recovery. Silhouette index

is used in the first case (unsupervised learning) and cluster information (supervised learning) is used in second case for the selection of p. In th third case, the distance coefficient is set to Euclidean and therefore p is set to 2.

TABLE 5.3: Comparision of three values of distance coefficient(p) in the purposed feature selection algorithms. The performance is evaluated using ARI index and are carried out in Gaussian mixed-model.

		Cluster Recovery (ARI) index [mean/std]					
		Silhoutee		Best Case		P equal 2	
		meanFSFW	maxFSFW	meanFSFW	maxFSFW	meanFSFW	maxFSFW
Cluster Recovery (ARI)	Noise Strength						
	No	0.68/0.22	0.915/0.09	0.954/0.04	**0.97**/0.03	0.94/0.05	0.959/0.04
	Half	0.685/0.25	0.862/0.15	**0.969**/0.03	0.935/0.08	0.958/0.04	0.874/0.12
	Full	0.729/0.23	0.81/0.17	**0.969**/0.03	0.939/0.08	0.958/0.05	0.817/0.17
	Noise Distribution						
	Gaussian	0.685/0.25	0.835/0.16	**0.968**/0.03	0.958/0.05	0.96/0.04	0.861/0.12
	Uniform	0.729/0.23	0.837/0.17	**0.97**/0.03	0.915/0.11	0.957/0.05	0.83/0.17
	Noise Model						
	NFA	0.821/0.18	0.958/0.05	**0.975**/0.03	**0.975**/0.03	0.962/0.05	0.941/0.07
	FB	0.629/0.28	0.902/0.14	**0.966**/0.03	**0.966**/0.03	0.964/0.03	0.947/0.08
	CFB	0.672/0.26	0.648/0.29	**0.966**/0.03	0.869/0.17	0.949/0.05	0.649/0.28
avg % of feature selected	Noise Strength						
	No	**26.4**/14.8	47.5/13.6	50.1/8.6	62.5/10.5	49/8.2	58.7/8
	Half	**29.9**/16.1	50.8/13.1	64.7/11.5	67/14.1	66.3/5.4	76.2/5.4
	Full	**32.9**/15	54.8/11.8	62.4/10.9	66.8/12.6	69.6/3.6	79.2/4.7
	Noise Distribution						
	Gaussian	**29.4**/15.4	51.8/13	62.2/11.4	67.9/13.2	65.6/5.2	76.3/5.4
	Uniform	**33.4**/15.7	53.8/12	64.9/11	65.9/13.4	70.2/3.8	79.1/4.7
	Noise Model						
	NFA	**31.8**/13.1	44.6/8	47.5/6.5	53.3/9.1	51.7/3.3	57.8/5.9
	FB	**32.7**/17.8	60.1/12.5	72.6/12.9	74.2/11.5	87.7/2.8	89.6/2.4
	CFB	**29.7**/15.7	53.7/17	70.5/14.2	73.1/19.4	64.4/7.4	85.7/6.9

– **Noise Strength**

With no noise added to the dataset it is observed that *max*-FSFW has better cluster recovery (ARI Index 0.97) than *mean*-FSFW (ARI Index=0.954). The performance of these algorithms in euclidean distance are close to the best cases. Selection of the distance coefficient using the Silhouette index greatly reduces the performance of *mean*-FSFW (ARI Index 0.68), with a high standard deviation (σ 0.22). However, *max*-FSFW performance does not deviate greatly (ARI Index 0.915), although dispersion is slightly higher (σ 0.09).

mean-FSFW gives better cluster recovery than *max*-FSFW in the best case (supervised), with the induction of half and full noise. A similar trend is observed in the performance of these algorithms with Euclidean distance. Moreover, selection of the distance coefficient gives better results with addition of noise using the Silhouette index in *mean*-FSFW. However, performance of *max*-FSFW decreases with the addition of

noise when the Silhouette index is used for selection the distance coefficient.

Feature space is highly reduced by *mean*-FSFW to 26.4%, 29.9% and 32.9% with no, half and full noise when p is selected using the Silhouette index. However, the next highest reduction is observed in *max*-FSFW which is increased by around 20% in all cases. *mean*-FSFW gives higher percentage of feature space reduction for all kind of noise strengths for selection of p values.

– **Noise Distribution**

In both type of noise distribution *mean*-FSFW gives better cluster recovery and higher feature space reduction than *max*-FSFW for the best case and Euclidean distance. Selection of p using the Silhouette index reverses the trend as *mean*-FSFW provides lower cluster recovery index with ARI 0.685 and 0.729 compared to *mean*-FSFW (ARI 0.835 and 0.837) for noise from Gaussian and Uniform distributions respectively. However, *mean*-FSFW is better at decreasing the feature space as feature space cardinality is reduced to 29.4% and 33.4 % compared to *max*-FSFW where feature space cardinality is decreased to 51.8% and 53.8 for noise from Gaussian and Uniform distributions respectively.

Overall, use of the Silhouette index has better reduction of feature space cardinality for both algorithms, followed by use of Euclidean distance and worst with best case (supervised)method.

– **Noise Model**

Both of these algorithms result higher cluster recovery with added noisy features under supervised conditions (ARI Index 0.975). Euclidean distance gives better results than the Silhouette index in both algorithms with added noise. A similar trend is observed in two other models of noise, feature bluring noise and cluster-based feature. Blurring noise with higher declination of cluster recovery in *mean*-FSFW as it dropped down to 0.629 and 0.672 respectively.

In terms of percentage of feature selection, *mean*-FSFW has better reduction of feature space with 31.8%, 32.7% and 29.7% of the feature space in the three noise models using the Silhouette index. Similarly, *max*-FSFW produces higher percentage of feature space reduction when using the Silhouette index.

5.3.6 UCI datasets

Thirteen different UCI datasets are taken from UCI machine learning repository (details of these data sets are given in Chapter 3, Section 3.9). As with the display of results from synthetic datasets, the three tables (5.1, 5.2, 5.3) below summarize the results from these real-world datasets. In the first table, feature selection algorithms are compared under the unsupervised condition where the Silhouette index is used for parameter tuning. In the second table, the best case for each algorithm is compared, whereas in the third table, the effect of distance coefficient on the performance of the proposed algorithms are presented.

Note:
Unlike synthetic datasets, each dataset from UCI repository have different configuration. Performance of *k*-Means types algorithms is comparatively higher in some dataset whereas too low in other. Therefore, higher standard deviation is observed when results from different datasets are combined together.

- **Parameter Tuning Using Silhouette Index**

In table 5.4 below, the performance of four different feature selection algorithms is compared with *k*-Means and *iMWk*-Means (no feature selection) using two metrics: ARI Index and the percentage of feature space selected.

TABLE 5.4: Comparision of four different feature selection algorithms with original variate of kMeans and intelligent Minkowski Weighted KMeans in real-world dataset with and without added noise where Silhoutee index is used for parameter tuning.

		Performance measurement [mean/std]						
		Without FS		With FS				
							FSFW	
		kmeans	iWkmeans	iKFS	FSFS	MCFS	meanFSFW	maxFSFW
avg cluster Recovery Index	Noise Strength							
	No	**0.557**/0.31	0.481/0.45	0.553/0.32	0.267/0.32	0.294/0.25	0.449/0.36	0.556/0.38
	Half	0.419/0.33	0.472/0.38	0.425/0.33	0.208/0.23	0.291/0.25	0.418/0.35	**0.496**/0.36
	Full	0.339/0.31	0.451/0.37	0.337/0.32	0.195/0.19	0.315/0.26	0.464/0.38	**0.492**/0.37
	Noise Distribution							
	Gaussian	0.403/0.32	0.458/0.37	0.411/0.32	0.221/0.24	0.293/0.24	0.451/0.36	**0.501**/0.35
	Uniform	0.355/0.33	0.465/0.38	0.352/0.33	0.181/0.17	0.312/0.27	0.431/0.37	**0.487**/0.38
	Noise Model							
	NFA	0.508/0.31	0.473/0.39	0.509/0.31	0.214/0.22	0.317/0.25	0.459/0.39	**0.538**/0.37
	FB	0.186/0.29	0.397/0.35	0.186/0.29	0.186/0.19	0.331/0.32	0.418/0.36	**0.473**/0.39
	CFB	0.442/0.28	**0.514**/0.38	0.449/0.28	0.204/0.21	0.26/0.17	0.446/0.36	0.47/0.35
avg % of feature selected	Noise Strength							
	No	-	-	88.2/26.7	19.6/19.3	**17.2**/24	33.7/14.8	52.9/31.5
	Half	-	-	91.3/19.6	15.8/16.6	**12.6**/12.6	24.9/15.1	53.2/27.6
	Full	-	-	91.4/16.2	17.2/17.7	**13.6**/15.7	28.9/18.4	59.6/29.8
	Noise Distribution							
	Gaussian	-	-	92.2/17.6	16/14.8	**11.8**/11.5	26/18.5	58.9/31.9
	Uniform	-	-	90.5/18.4	17/19.3	**14.5**/16.5	27.8/15.2	54/25.4
	Noise Model							
	NFA	-	-	96.9/7.2	8.6/5.4	**6.6**/5	22.3/11.6	51.8/25.8
	FB	-	-	86.5/21	16.4/13.8	**16**/16.3	31.1/20.1	57.4/30.9
	CFB	-	-	90.6/20.8	24.5/23.3	**16.9**/16.1	27.3/17.1	60.1/29.5

– **Noise Strength**

In the real-world datasets, *k*-Means provides best cluster recovery with ARI Index equal to 0.577 in the case of no noise. A similar performance is achieved in *max*-FSFW (ARI Index 0.556) with feature space cardinality reduced to 52.9%. The next best cluster recovery is obtained by IKFS (ARI Index 0.553) although its contribution toward feature space cardinality is minimal (88.2%). Among the feature selection algorithms, second best cluster recovery is achieved by *mean*-FSFW (ARI Index 0.449), with reduction of feature space cardinality to 33.7%. However, the best feature space reduction

is observed in MCFS (17.2%) but it provides low cluster recovery (ARI 0.294).

max-FSFW shows a better cluster recovery than *k*-Means and *iMWk*-Means with half and full strength noise as *max*-FSFW has ARI Indexes equal to 0.496 and 0.492 respectively. In the case of half noise, the second best cluster recovery is observed in *iWk*-Means with ARI Index 0.472. Among the feature selection algorithms, *mean*-FSFW gives second best results. Moreover, with increasing noise strength, cluster recovery in all algorithms (with and without feature selection) are decreased except for *mean*-FSFW. In terms of reduction of feature space cardinality, MCFS gives better performance, followed by FSFS, *mean*-FSFW and *max*-FSFW.

– **Noise Distribution**

A similar trend is observed with uniform noise added in the real-world datasets where *max*-FSFW provides best cluster recovery with ARI index equals to 0.487%, followed by *mean*-FSFW (ARI index 0.431%), with the lowest cluster recovery in FSFS (ARI index 0.181). In Gaussian noise, *max*-FSFW has best cluster recovery (ARI 0.501), followed by *iWk*-Means (ARI 0.458) and finally FSFS (ARI 0.411). MCFS shows best reduction of feature space cardinality (8.8%), followed by FSFS(12%). MCFS produces the best feature reduction for both Gaussian (11.8%) and Uniform (14.5%) noise distributions, closely followed by FSFS.

As observed in the synthetic datasets, IKFS shows no contribution to reduce feature space cardinality in real-world datasets. In both types of noise distribution, IKFS only reduces feature space cardinality by less than 5%.

– **Noise Model**

max-FSFW provides better cluster recovery (ARI Index 0.958) than *k*-Means and its weighted variant *Wk*-Means with the addition of noisy features. Interestingly, in this configuration, deviations of the ARI indexes are relatively low as STD of these values results to 0.05. The next closest performer among the feature selection algorithms is *mean*-FSFW with ARI index equals to 0.821; the performance of iKFS is not considered since it does not contribute to reduction of feature space cardinality. In term of reduction of feature space cardinality, MCFS provides best result with 4.8% of feature selection; however, MCFS gives a much lower cluster recovery (ARI Index equals to 0.414) than the maxFSFS.

Feature blurring with noise, *max*-FSFW provides best cluster recovery with ARI Index equal to 0.902 followed by the weighted variant of *k*-Means, *Wk*-Means. Though FSFS shows a high percentage of reduction of feature space cardinality, with the average percentage of feature selection equal to 9.1%, it gives relatively low cluster recovery (ARI index equals to 0.137) among the three selected feature selection algorithms.

Again, with cluster-based feature blurring noise added, cluster recovery is relatively lower in *max*-FSFW than in the previous two noise models and the weighted variant of *k*-Means, *Wk*-Means. In the configuration, *mean*-FSFW produces better cluster recovery (average ARI Index equal to 0.672) among the feature selection algorithms considered. In respect to reduction of feature space cardinality, MCFS provides best results, with 11.4% of features selected, followed by FSFS, *mean*-FSFW and *max*-FSFW.

Overall, it is concluded that *max*-FSFW gives best performance (better than *k*-Means and *Wk*-Means) with almost 50% reduction of feature space cardinality when datasets have added noisy features or feature blurring noise.

- **Best Case**

In table 5.5 below, the best possible results in the real-world datasets from each feature selection algorithm are compared with those from *k*-Means and *iWk*-Means.

TABLE 5.5: Comparison of four different feature selection algorithms with original variate of KMeans and intelligent Minkowski Weighted KMeans in real-world dataset under the supervised condition (best possible cases).

		Performance Measurement (mean/std)						
					Best Possible Cases			
		kMean	iWKMeans	iKFS	FSFS	MCFS	meanFSFW	maxFSFW
avg cluster Recovery Index	Noise Strength							
	No	0.557/0.31	0.481/0.45	0.553/0.32	**0.615**/0.3	0.614/0.29	0.571/0.39	0.601/0.34
	Half	0.419/0.33	0.472/0.38	0.425/0.33	0.55/0.28	0.561/0.28	**0.612**/0.31	0.578/0.31
	Full	0.339/0.31	0.451/0.37	0.337/0.32	0.546/0.27	0.544/0.31	**0.612**/0.31	0.591/0.31
	Noise Distribution							
	Gaussian	0.403/0.4	0.458/0.46	0.411/0.41	0.575/0.58	0.548/0.55	**0.601**/0.6	0.576/0.58
	Uniform	0.355/0.35	0.465/0.47	0.352/0.35	0.521/0.52	0.557/0.56	**0.622**/0.62	0.593/0.59
	Noise Model							
	NFA	0.508/0.31	0.473/0.39	0.509/0.31	0.571/0.28	0.585/0.28	**0.588**/0.35	0.58/0.33
	FB	0.186/0.29	0.397/0.35	0.186/0.29	0.491/0.28	0.511/0.34	**0.605**/0.3	0.581/0.31
	CFB	0.442/0.28	0.514/0.38	0.449/0.28	0.582/0.26	0.561/0.26	**0.643**/0.28	0.592/0.29
avg % of feature selected	Noise Strength							
	No	-	-	88.2/26.7	19.6/10.4	**17.2**/12.5	33.7/20.4	52.9/15.7
	Half	-	-	91.3/19.6	15.8/10.4	**12.6**/5.7	24.9/12.3	53.2/24.6
	Full	-	-	91.4/16.2	17.2/10.3	**13.6**/4.4	28.9/10.2	59.6/20.4
	Noise Distribution							
	Gaussian	-	-	92.2/15.7	16	**11.8**/5.6	26/12.4	58.9/24.5
	Uniform	-	-	90.5	17/18.2	**14.5**/6.7	27.8/10.5	54 /22.3
	Noise Model							
	NFA	-	-	96.9/7.2	8.6/2.4	**6.6**/3.4	22.3/12.4	51.8/24.8
	FB	-	-	86.5/21	16.4	**16**/10.4	31.1	57.4/25.9
	CFB	-	-	90.6/20.8	24.5	**16.9**/12.5	27.3	60.1/30.6

– **Noise Strength**

Under supervised learning (cluster label provided), FSFS shows the best cluster recovery with average ARI Index equals to 0.615, closely followed by MCFS (avg ARI Index equals to 0.614) and *max*-FSFW (avg ARI Index equals to 0.571). Feature selection algorithms show better cluster recovery under supervised learning when no noise is added in the real-world datasets. In terms of percentage of feature selection, MCFS has better reduction on feature space cardinality, with 17.2% of features selection.

mean-FSFW shows best cluster recovery in the case of both half and full added noise, with average ARI Index equal to 0.612, closely followed by *max*-FSFW. In the case

of half and full noise, all feature selection algorithms including MCFS and FSFS provides better cluster recovery than *k*-Means and *iWk*-Means. In terms of reduction of feature space cardinality, MCFS produces best result for all noise type, followed by FSFS.

- **Noise Distribution**

  In both Gaussian and Uniform noise distribution, *mean*-FSFW provides best cluster recovery, with ARI Indexes reading 0.601 and 0.622 followed by *mean*-FSFW. In terms of reduction of feature space cardinality, MCFS produces the best outcome as it is able to reduce feature space cardinality to 11.8% and 14.5% for noise from Gaussian and Uniform distribution.

- **Noise Model**

  mean-FSFW shows better cluster recovery in all three types of noise models: added noisy features, feature-blurring noise and cluster-based feature blur noise. On the basis of the percentage reduction in feature space cardinality, MCFS gives the best results, with feature space cardinality reduced to 6.6%, 16% and 16.9% respectively under the supervised learning environment.

- **Distance coefficient**

  In table 5.6 below, the performance of two proposed algorithms is compared against that from three methods of selecting the distance coefficient in real-world datasets. In the first and second cases, p is selected from a range of 1.1 to 5.0 with interval of 0.1, which maximized cluster recovery. The Silhouette index is used in the first case (unsupervised learning) and cluster information (supervised learning) is used in the second case for the selection of the p. In third case, the distance coefficient is set to Euclidean and therefore p is set to 2.

TABLE 5.6: Comparision of the purposed feature selection algoirthm with three different values of distance coefficient(p). The performance is evaluated using ARI index and are carried out in Gaussian mixed-model.

		Cluster Recovery (ARI) index [mean/std]					
		Silhoutee		Best Case		P equal 2	
		meanFSFW	maxFSFW	meanFSFW	maxFSFW	meanFSFW	maxFSFW
Cluster Recovery (ARI)	Noise Strength						
	No	0.449/0.36	0.556/0.38	0.571/0.39	**0.601**/0.34	0.45/0.47	0.535/0.38
	Half	0.418/0.35	0.496/0.36	**0.612**/0.31	0.578/0.31	0.498/0.37	0.512/0.32
	Full	0.464/0.38	0.492/0.37	**0.612**/0.31	0.591/0.31	0.496/0.35	0.476/0.32
	Noise Distribution						
	Gaussian	0.451/0.36	0.501/0.35	**0.601**/0.32	0.576/0.3	0.499/0.36	0.516/0.31
	Uniform	0.431/0.37	0.487/0.38	**0.622**/0.31	0.593/0.31	0.495/0.36	0.472/0.32
	Noise Model						
	NFA	0.459/0.39	0.538/0.37	**0.588**/0.35	0.58/0.33	0.495/0.39	0.526/0.33
	FB	0.418/0.36	0.473/0.39	**0.605**/0.3	0.581/0.31	0.511/0.34	0.482/0.32
	CFB	0.446/0.36	0.47/0.35	**0.643**/0.28	0.592/0.29	0.484/0.36	0.473/0.31
avg % of feature selected	Noise Strength						
	No	**33.7**/14.8	52.9/31.5	42.1/15	72.1/29.6	46.1/9.3	76.8/20.3
	Half	**24.9**/15.1	53.2/27.6	49.8/22.9	72.1/24.1	54.4/14.2	82.3/15.6
	Full	**28.9**/18.4	59.6/29.8	56.1/24.5	76/22.9	58/18.2	83.1/15.7
	Noise Distribution						
	Gaussian	**26**/18.5	58.9/31.9	52/24	77.1/21.7	56.2/18	82.7/14.7
	Uniform	**27.8**/15.2	54/25.4	53.9/23.8	71/25	56.2/14.7	82.7/16.6
	Noise Model						
	NFA	**22.3**/11.6	51.8/25.8	40.1/15.5	65/21.7	44.5/12.1	77.1/18.2
	FB	**31.1**/20.1	57.4/30.9	63.3/27	75.2/22.2	66.3/15.4	85.1/12.6
	CFB	**27.3**/17.1	60.1/29.5	55.5/21.8	81.9/24.1	57.7/13.7	85.9/14.2

– **Noise Strength**

Silhouette index results in better cluster recovery with *max*-FSFW in all three cases of noise strength, whereas the performance is reversed in the case of feature space reduction. In the best case, *max*-FSFW provides the better cluster recovery (ARI 0.601) with no noise, while with added noise *mean*-FSFW provides better cluster recovery but *mean*-FSFW results in better reduction of feature space for all noise strengths.

max-FSFW produces better cluster recovery than *mean*-FSFW for no and half noise strengths and *max*-FSFW gives better cluster recovery for full noise strength when Euclidean distance is used. In term of feature space reduction, *mean*-FSFW provides better result for all cases of noise strength.

– **Noise Distribution**

In both noise distribution, *max*-FSFW gives better cluster recovery than *mean*-FSFW for the Silhouette index and is reverse in best case. For Euclidean distance, *max*-FSFW has better cluster recovery (ARI 0.516), *mean*-FSFW gives better cluster recovery (ARI 0.495) for Gaussian and Uniform noise distribution.

In term of feature space cardinality, *mean*-FSFW is able to reduce the feature space better than *mean*-FSFW, with best result using the Silhoutte index.

– **Noise Model**

max-FSFW provides better cluster recovery than *mean*-FSFW for Silhouette index and Euclidean distance when noisy features are added to the datasets, and performance is reversed in the best case. Blurring the feature with noise, FB and CFB, *mean*-FSFW provides better cluster recovery than *max*-FSFW in Silhouette index and Euclidean distance, and this is reversed in the best case.

In terms of percentages of feature selection, *mean*-FSFW is able to reduce the feature space better than *max*-FSFW, with best result when using the Silhouette index in all cases of noise model.

5.4 Conclusion

From the above study, the proposed feature weighting based feature selection algorithm appears outperform the other two feature selection algorithms considered, FSFS and MCFS, in both the synthetic and real world datasets. IKFS performs worst, with no contribution to feature selection. Under the supervised condition (best case), *mean*-FSFW and *max*-FSFW are both perform better than the benchmark algorithms, *k*-Means, and *iWk*-Means in both the synthetic and real-world datasets, for all noise configurations . Under the unsupervised condition, *mean*-FSFW and *max*-FSFW provides better cluster recovery than *k*-Means and *iWk*-Means when datasets have noise.

max-FSFW is slightly better than *mean*-FSFW in term of cluster recovery. However, in terms of feature space reduction, *mean*-FSFW is better than *max*-FSFW. Either *mean*-FSFW or *max*-FSFW can be used as feature selection technique depending on the objectives of the experiment.

In overall, both the proposed feature selection algorithms found be better in term of reduction of feature space with maximizing the cluster structure in unsupervised environment. However, the proposed feature selection algorithms are computationally expensive as they are required to find appropriate distance coefficients. But this can be overcome under semi-supervised learning where appropriate distance coefficient can be identified by running few sample tests in a small dataset.

Chapter 6

Prediction of Missing Values

Raw data is likely to have missing values[160]. They are the result of technical fault or human negligence during data collection, data entry or pre-processing period, or lack of availability of data. Data, collected from experiment or survey might have some data values unobserved. Attribute values can be unobserved under various reasons. For example, in a household survey, an individual might refuse to disclose their income if his/her income is low. In industrial experiments, mechanical failure is one of the main reasons for the loss of data values. Similarly, respondents may not be able to set their preferred option/s from multiple choice questionable in the survey. In such scenarios, respondents may refuse to answer the questions. However, most data processing tools expect dataset to have no missing values.

6.1 Aims

In this chapter, three different approaches: imputation, Column-wise deletion and partial distance are used to address missing values in *k*-Means type algorithms. Four different imputation methods analysed in this experiment are replacing the missing values by: feature average, KNN imputation, simple regression and cluster-based regression. These methods are tested with weighted *k*-Means (*Wk*-Means) and intelligent weighed *k*-Means (*iWk*-Means). Column-wise deletion approach removes the feature containing missing values before the data are clustered. In Partial distance approach, the distance metric within the clustering algorithm is modified to address the missing values. Partial distance has been applied in *k*-Means and its weighted variant, *Wk*-Means. The missing values used in this research are missing completely at random (MCAR), and not missing at random (NMAR) mechanism [see section 3.8.2]. Moreover, the missing values implanted for test purposes are limited to two cases of univariate missing pattern: missing values in a feature which has highest and lowest correlation with its dataset. These methods are tested in both real-world datasets from UCI machine learning repository and synthetic datasets, generated from mixed-model Gaussian distribution.

The aims of this chapter are to observe:

- **Different imputation methods in *Wk*-Means and *iWk*-Means**
  In the first half of the chapter, impacts of four different imputation methods on the performance of *Wk*-Means and *iWk*-Means is observed. Two of these methods are based on average imputation: replacing the missing values with an attribute mean and replacing the missing values by an average of 'k' nearest values, KNN imputation. Two other remaining imputation methods are based on values derived by regressions: from the whole dataset and the cluster to which the missing value belongs.

- **The effect of correlation of feature on cluster recovery approaches**
  In univariate missing pattern missing values are present in only one of the features. The correlation between features that posses missing values with the other remaining features

has an impact on the choice of imputation methods. In attempting to find the influence of correlation, univariate missing patterns are checked with the features that have highest and lowest correlation with their datasets.

- **The impact of two missing mechanisms on cluster recovery approaches**
  As stated earlier, the experimental datasets have missing values from missing completely at random (MCA) and not missing at random (NMAR) mechanism. One of the area of this research is to find how missing values from different mechanisms influence the cluster recovery process i.e. performance of *Wk*-Means and *iWk*-Means.

- **How three common approaches - imputation, column-wise data deletion and partial distance influence clustering**
  There are three common practices while dealing with missing values: Imputation, case deletion and redefining the distance matrix. In this research, we observed how these three different approaches influence cluster recovery in *k*-Means and *Wk*-Means.

6.2 Experimental Set-up

6.2.1 Setting up Dataset

Simulations for the missing values experiment are carried in both synthetic and the real-world datasets. The synthetic datasets, used earlier in Chapter 5, are used in this chapter. Therefore, we have four configurations of Gaussian mixed model drawn from feature set V={8, 12,16, 20} and a set of number of mix component K={2 ,3, 4, 5}. For each configuration we have 20 datasets and each dataset contains 1000 entities. Configuration of datasets in the real-world varies and are distinct from each other. Therefore, each dataset from real-world is considered as a unit configuration. Seven datasets from UCI machine learning repository are used for the real-world experiment.

6.2.2 Addition of Missing Values

In our experiment, univariate missing patterns in both synthetic and real-world datasets are used. One of the features is selected based on the linear correlation between the feature and its dataset. Two cases of correlation which are considered for the selection are:

1. highest correlation,
2. lowest correlation

Note:
For a dataset Y is defined in a feature space V = $\{v_1, v_2, v_3, v_4\}$, the correlation of a feature, v_1 with its dataset, Y is computed as linear correlation between v_1 and dataset Y_{prime}=Y-v_1 => $\{v_2, v_3, v_4\}$.

Moreover, we examined two missing mechanism for above cases:

- missing completely at random(MCR) , and
- not missing at random (NMAR).

Two sets of experiments are run for each missing mechanism. For the first set, a feature with highest correlation with its dataset is chosen and for the second set, the feature with lowest correlation with its dataset are selected.

In each of the above cases, experiments are conducted with different percentage of missing values. The missing percentage of data in the feature is set to 5%, 10%, 15%, 20%, 25% , 30% and 35%.

Once the target feature and number of missing entities are derived, the original values are plugged out from the experimental datasets in random fashion for MCR mechanism. For NMAR, we randomly plug out values which are greater than the feature mean value. Since we have used a mean value as an index for NMAR, the maximum number of missing values in NMAR is limited to number of values greater than the feature mean value.

6.2.3 Selection of Algorithms

In this experiments, three general approaches are observed to deal with missing values:

- Imputation method
  Four different Imputation methods: feature average, average of K nearest neighbours, simple regression and cluster-based regression are tested.
- Removing the feature that have missing values
  Another approach to deal with missing value is to remove the feature or entity which has missing values. In our experiment, we are dealing with only univariate missing values; therefore to deal with the missing values, we have deleted the feature that contain missing values
- Partial distance
  One of the possible method to avoid imputation for dealing with missing values is to modify the distance metric inside the algorithm itself. In our experimental set, we modified the distance matrix so that an average distance from the known feature is considered between two entities to measure the similarity matrix. The same approach is carried out while operating with centroids: redefining the centroids for the cluster that have missing values and measuring the similarity between centroids and the entity which has missing value.

A set of experiment for different imputation methods is used the earlier section of this chapter. A new set of experiment is carried in the later section, to compare three different approaches to deal with missing values.

6.2.4 Extension to Noise

To observe the effect of noise over the performance of all cases of handling the missing values as discussed above, we have extended our experiment with addition of $m/2$ and m numbers of noisy features. Similar to the chapter 5, noises are drawn from uniform distribution (white noise).

6.3 Reading Results

All the experimental results are displayed in a tabular form. Two factors need to consider while reading the results:

- Organization of results
- Heading of tables.

The two sections below explain how results are organised and how to read the data tables.

6.3.1 Organization of Results

Results from the experiments are grouped into two categories:

- comparison between different imputation methods and
- comparison between different approaches.

For each categories above, there are two tables containing results where missing values are in the feature with:

- minimum and
- maximum

correlation with its dataset.

In each table, results are further summarised based on:

- dataset configurations,
- percentage of missing values and
- types of added noise.

A horizontal divider is placed along the columns in each table that divides each table into the three observations listed above.

We have used two set of tables to display results from imputation methods and to compare results from different approaches.

Results for Imputation Methods

The tables that display results from imputation methods have nine columns. The first column displays the configuration of the experimental set-up which is either dataset configurations, percentage of missing values or type of noise. Columns from two to five display ARIs for *Wk*-Means which are obtained after replacing the missing values with feature average, mean of K nearest values, values derived from regression on the whole dataset and the value derived from regression on the cluster in which the particular entity with missing values belongs to, respectively. The same pattern is set to display results from *iWk*-Means from columns six to nine.

Results from Different Approaches

Tables which are used to evaluate (compare) the partial distance approach with feature mean imputation and feature removal approach have seven columns. As in earlier case, the first column contains information about datasets, missing values percentage or noise type. The second and third columns contain ARIs of *k*-Means and *Wk*-Means where missing values are replaced by the feature average. The fourth and fifth columns contain ARIs of *k*-Means and *Wk*-Means when the feature containing the missing values is removed and finally, the last two columns have ARIs of *k*-Means and *Wk*-Means where distance matrix within the clustering algorithms are replaced by partial distance.

6.4 Experimental Results

Experimental results, derived from comparison of four different imputation methods and three different approaches, are explained in 6.4.1 and 6.4.2.

6.4.1 Imputation of Different Algorithms for Weighted *k*-Means and Intelligent Weighted *k*-Means

Imputation, i.e. replacement of missing value by known value is one of the common approach for resolving the missing values problem during data pre-processing. Four different imputation methods: feature average, K-nearest neighbours, simple regression and cluster-based regression are observed in *Wk*-Means and *iWk*-Means . Each of these methods are tested in two different missing mechanisms: missing completely at random (MCAR) and not missing at random (NMAR) mechanism in sub-section 6.4.1 and 6.4.1 respectively.

Missing values with MCAR Mechanism

In this section, missing completely at random mechanism is observed in synthetic and real-world datasets.

A) **Synthetic datasets**
For two cases of univariate missing pattern, i.e. missing values in a feature with minimum correlation and a feature with maximum correlation, experimental results from four Gaussian mixed model configuration with MCAR mechanism are explained in this section.

A.i) **Univariate missing pattern with minimum correlation**

In table 6.1, ARI values for *Wk*-Means and *iWk*-Means are listed from the synthetic datasets with MCAR missing values in a feature with minimum correlation with its dataset. Results are categorized based on datasets, percentage of missing values and noise types.

TABLE 6.1: Comparison of four different imputation methods in Gaussian mixed model where missing values have MCAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset.

MINIMUM CORRELATION		*Wk*-Means				Intelligent *Wk*-Means			
		F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
	Dataset	three different cardinality of Gaussian clusters with covariance matrix equal to 0.5							
	1000x8	**0.971/0.04**	**0.971/0.04**	**0.971/0.04**	**0.971/0.04**	**0.197/0.06**	0.196/0.05	0.195/0.05	0.195/0.05
	1000x12	**0.922/0.18**	**0.922/0.18**	**0.922/0.18**	**0.922/0.18**	0.372/0.13	0.377/0.13	0.376/0.13	**0.378/0.13**
	1000x16	**0.881/0.17**	**0.881/0.17**	**0.881/0.17**	**0.881/0.17**	0.624/0.12	**0.625/0.12**	0.624/0.12	**0.625/0.12**
	1000x20	**0.856/0.15**	**0.856/0.15**	**0.856/0.15**	0.855/0.15	**0.85/0.09**	0.849/0.09	0.849/0.09	0.848/0.09
	Missing %	Evaluation based on % of missing values							
	1	**0.911/0.15**	**0.911/0.15**	**0.911/0.15**	**0.911/0.15**	**0.514/0.27**	0.513/0.27	0.513/0.27	0.513/0.27
	10	**0.905/0.15**	0.904/0.16	0.903/0.16	0.903/0.16	0.511/0.27	0.512/0.27	0.511/0.27	**0.513/0.27**
	15	**0.905/0.15**	**0.905/0.15**	**0.905/0.15**	**0.905/0.15**	0.511/0.27	**0.512/0.27**	0.511/0.27	**0.512/0.27**
	20	0.905/0.15	**0.906/0.15**	**0.906/0.15**	**0.906/0.15**	0.509/0.27	0.511/0.27	**0.512/0.27**	**0.512/0.27**
	25	0.91/0.15	0.91/0.15	**0.911/0.15**	0.91/0.15	0.509/0.27	0.51/0.27	0.51/0.27	**0.511/0.27**
	30	**0.909/0.15**	0.908/0.15	0.908/0.15	0.908/0.15	0.51/0.27	**0.511/0.27**	0.51/0.27	**0.511/0.27**
	35	**0.908/0.15**	0.907/0.15	**0.908/0.15**	0.907/0.15	**0.513/0.26**	**0.513/0.27**	0.509/0.27	0.511/0.27
	Noise Type	Performance of the imputation methods with added uniform noisy features							
	No Noise	**0.908/0.15**	0.907/0.15	**0.908/0.15**	0.907/0.15	0.511/0.27	**0.512/0.27**	0.511/0.27	**0.512/0.27**
	Half Noise	0.856/0.23	**0.857/0.23**	0.856/0.23	**0.857/0.23**	0.795/0.24	**0.796/0.24**	0.795/0.24	**0.796/0.24**
	Full Noise	0.821/0.21	0.822/0.21	0.822/0.21	**0.823/0.2**	**0.822/0.18**	**0.822/0.18**	**0.822/0.18**	**0.822/0.18**

- Dataset Configuration
  Weighted *k*-Means does not show any difference for the synthetic datasets when missing values replaced using any of the four algorithms. The average cluster recovery for four dataset configurations read: 0.971, 0.922, 0.881 and 0.856. However, increment of the cardinality of feature space decreases performance of *Wk*-Means with the same amount for all imputation methods.

In case of Intelligent *Wk*-Means, mixed results are observed as no imputation method shows clearly the best choice. In two dataset configurations with feature space cardinality of 8 and 20, replacing missing values with feature average has better cluster recovery in *iWk*-Means. Likely, KNN imputation method has shown better cluster recovery for datasets with feature space cardinality 16 and cluster-based regression shows better cluster recovery in *iWk*-Means for datasets with feature space cardinality 12 and 16. Simple regression based imputation method gives lower performance for *iWk*-Means.

- Percentage of missing values
  In *Wk*-Means, replacing the missing values with feature average and simple regression don't follow the trend of changes with respect to the percentage of missing values. However, replacing the missing values with KNN and cluster-based regression are comparatively better when the percentage of missing values is lower for *Wk*-Means. In case of *iWk*-Means, none of four methods show any trend of changes with respect to percentage of missing values.

- Type of noise
  With the increment of noisy features, the performance of *Wk*-Means decreases for all imputation methods but for IKWMeans performance gets better when the number of noisy features increase. *Wk*-Means has better performance when missing values are replace with feature average or simple regression whereas, cluster-based regression is best alternative for noisy datasets. Imputation of missing values with KNN or cluster-based regression is the best choise for intelligent *Wk*-Means in any types of noise.

A.ii) **Univariate missing pattern with maximum correlation**

In table 6.2, four different imputation methods - feature average, KNN, simple regression and cluster-based regression are observed based on their effect on the performance of *Wk*-Means and Intelligent Weighted *k*-Means. Experiments are carried out in synthetic datasets which have mixed-model Gaussian distribution with well separated Gaussian clusters (cluster covariance equals to 0.5). The missing values have an univariate missing pattern and are located in the feature which have highest correlation with its dataset. The missing values are generated using 'missing completely at random' mechanism.

We further analysed the results based on dataset configurations, percentages of missing values and noise types. The best performance of each observation (row-wise) is highlighted with bold letter. Each entity in the performance represents the average value of the reading and the standard deviation of the observation separated by forward slash(/).

- Dataset Configurations
  Replacing the missing values with a feature average gives better cluster recovery in *Wk*-Means when the missing value are present in a feature which have maximum correlation with its dataset. Under this configuration, for datasets, with smaller feature space cardinality(eight), replacing missing values with KNN gives better cluster recovery for *Wk*-Means. Similarly, KNN is the best option to replace the missing values for intelligent *Wk*-Means. However, in larger feature space cardinality (20), cluster-based regression method is the best to replace the

TABLE 6.2: Comparison of four different imputation methods in Gaussian mixed models where missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has highest correlation with its dataset.

		WKMeans				Intelligent *Wk*-Means			
		F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
MAXIMUM CORRELATION	Dataset	Three different cardinality of Gaussian clusters with covariance matrix equal to 0.5							
	1000x8	0.951/0.06	**0.954/0.06**	0.95/0.06	0.951/0.06	0.164/0.05	**0.231/0.07**	0.192/0.05	0.218/0.06
	1000x12	0.92/0.17	0.918/0.18	**0.921/0.17**	0.917/0.18	0.334/0.11	**0.408/0.12**	0.356/0.1	0.402/0.12
	1000x16	**0.886/0.17**	0.884/0.17	0.885/0.17	0.883/0.17	0.576/0.11	**0.636/0.12**	0.602/0.12	0.635/0.12
	1000x20	**0.857/0.15**	0.856/0.15	**0.857/0.15**	0.856/0.15	0.834/0.1	0.838/0.09	0.833/0.1	**0.844/0.09**
	Missing %	Evaluation based on % of missing values							
	1	**0.904/0.16**	**0.904/0.16**	**0.904/0.16**	**0.904/0.16**	0.505/0.27	**0.514/0.27**	0.512/0.27	**0.514/0.27**
	10	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	0.905/0.15	0.484/0.27	0.518/0.26	0.503/0.27	**0.52/0.26**
	15	0.906/0.15	0.906/0.15	**0.907/0.15**	0.905/0.15	0.474/0.27	0.518/0.26	0.497/0.26	**0.523/0.26**
	20	**0.904/0.15**	0.902/0.15	**0.904/0.15**	0.901/0.15	0.471/0.27	0.524/0.25	0.493/0.26	**0.525/0.26**
	25	0.902/0.15	**0.903/0.15**	**0.903/0.15**	0.9/0.15	0.468/0.27	**0.53/0.24**	0.49/0.26	0.528/0.25
	30	0.901/0.15	0.901/0.15	**0.902/0.15**	0.899/0.15	0.467/0.27	**0.54/0.24**	0.488/0.26	0.53/0.25
	35	**0.901/0.15**	0.898/0.15	0.899/0.15	0.897/0.15	0.469/0.27	**0.553/0.23**	0.486/0.26	0.535/0.25
	Noise Type	Performance of the imputation methods with added uniform noisy features							
	No Noise	**0.904/0.15**	0.903/0.15	**0.904/0.15**	0.902/0.15	0.477/0.27	**0.528/0.25**	0.496/0.26	0.525/0.26
	Half Noise	0.843/0.24	0.85/0.24	**0.852/0.23**	0.85/0.23	0.763/0.26	0.793/0.23	0.786/0.24	**0.799/0.23**
	Full Noise	0.809/0.21	0.814/0.21	**0.819/0.21**	0.818/0.21	0.791/0.21	0.812/0.19	0.814/0.19	**0.823/0.18**

missing values for intelligent *Wk*-Means.

- Percentage of missing values
  All four imputation methods have a similar effect on the performance of *Wk*-Means when the percentage of missing value is relatively low (less than or equal to 10). Replacing the missing value with a simple regression method is the best option for *Wk*-Means when the percentage of missing values is less than or equal to 30%. In case of relatively large number of missing values(around 35%), feature average imputation results better cluster recovery for *Wk*-Means.

  For intelligent weighted *k*-Means, replacing the missing values with cluster-based regression is a better option when number of missing features are small (less than 20%) whereas, the number of missing values are relatively high in KNN imputation which is the best option.

- Type of noise
  When there are no noisy features in the datasets, feature average and cluster-based regression are the best imputation methods for *Wk*-Means. Replacing the missing values with simple regression is the optimal option for *Wk*-Means when noisy features are added. Simple imputation performances best in *iWk*-Means with no noisy features in datasets while cluster-based regression method is better with the additional noisy features.

Classification of performance is based on the percentage missing, no significant trend is noticed for *iWk*-Means in case of maximum and minimum correlation. However, with addition of noisy features, cluster-based regression perform better in *Wk*-Means and *iWk*-Means for both sets of experiment(missing values in maximum and minimum correlated feature).

B) **Real-world datasets**

Similar to the synthetic datasets, for two cases of univariate missing pattern discussed earlier, experimental results from seven different real-world datasets from UCI machine learning repository are examined with MCAR mechanism in this section.

B.i) **Univariate missing pattern with minimum correlation**
In table 6.3, four different imputation methods - feature average, KNN, simple regression and cluster-based regression are observed based on their effect on the performance of *Wk*-Means and *iWk*-Means in the real-world datasets. For the case study, missing values are presented for the feature with minimum correlation and its dataset.

TABLE 6.3: Comparison of four different imputation methods in the real-world datasets where missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset.

MINIMUM CORRELATION

	Wk-Means				Intelligent *Wk*-Means			
	F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
dataset	Seven different real world databases							
Ecoli	0.19/0.19	0.194/0.19	0.205/0.19	**0.21/0.2**	**0.038/0**	**0.038/0**	**0.038/0**	**0.038/0**
Glass	**0.187/0.05**	0.185/0.05	0.186/0.05	0.17/0.05	0.138/0.07	0.138/0.07	0.128/0.06	**0.154/0.07**
Heart	0.176/0.11	0.176/0.11	**0.177/0.11**	**0.177/0.11**	**0.176/0**	**0.176/0**	**0.176/0**	**0.176/0**
Iris	**0.724/0.18**	0.718/0.17	**0.724/0.15**	0.715/0.17	0.618/0.02	0.63/0.02	0.619/0.02	**0.631/0.02**
Soya	0.63/0.17	0.619/0.18	0.633/0.19	**0.645/0.2**	**0.943/0.06**	0.844/0.15	0.927/0.09	0.898/0.12
Wine	0.787/0.05	0.787/0.05	**0.79/0.03**	0.788/0.05	0.628/0.01	0.63/0.02	**0.633/0.02**	**0.633/0.02**
Zoo	0.587/0.16	0.587/0.16	0.586/0.17	**0.593/0.15**	**0.907/0**	**0.907/0**	0.905/0.02	0.899/0.03
Missing %	evaluation based on % of missing values							
1	0.459/0.28	0.455/0.27	0.459/0.28	**0.464/0.28**	**0.476/0.36**	0.465/0.35	0.464/0.35	0.457/0.35
10	0.448/0.28	0.454/0.29	**0.457/0.29**	0.454/0.3	**0.486/0.35**	0.48/0.34	0.483/0.35	0.482/0.34
15	**0.489/0.3**	0.483/0.3	0.475/0.29	0.468/0.29	0.494/0.35	0.476/0.34	0.491/0.35	**0.497/0.34**
20	0.468/0.29	0.475/0.29	**0.476/0.29**	**0.476/0.3**	**0.498/0.35**	0.481/0.33	0.497/0.35	**0.498/0.34**
25	0.489/0.29	0.486/0.29	0.489/0.29	**0.501/0.3**	0.496/0.35	0.486/0.33	0.494/0.35	**0.498/0.34**
30	0.461/0.28	0.45/0.28	**0.476/0.29**	0.462/0.28	**0.499/0.35**	0.479/0.33	0.498/0.35	**0.499/0.35**
35	0.468/0.3	0.463/0.29	0.469/0.29	**0.472/0.3**	**0.499/0.35**	0.495/0.34	**0.499/0.34**	0.497/0.34
Noise Type	Performance of the imputation methods with added uniform noisy features							
No Noise	0.469/0.29	0.467/0.29	**0.472/0.29**	0.471/0.29	**0.493/0.35**	0.48/0.34	0.489/0.35	0.49/0.34
Half Noise	0.475/0.25	0.483/0.26	0.481/0.25	**0.485/0.25**	**0.475/0.34**	0.463/0.33	0.47/0.34	0.452/0.32
Full Noise	0.456/0.25	0.455/0.25	0.46/0.25	**0.463/0.25**	**0.424/0.31**	0.401/0.31	0.421/0.31	0.409/0.3

- datasets
  Ecoli dataset shows best cluster recovery with 0.21 ARI value for *Wk*-Means when missing values are replaced by cluster-based regression. In case of *iWk*-Means, ARIs are relatively low (0.038) for all imputation methods.

  For the Glass dataset, replacing the missing values with feature average results better cluster recovery for *Wk*-Means whereas replacing the missing values with cluster-based regression has the worst effect. However, for IWKMens, replacing the missing values with cluster-based regression have better cluster recovery (ARI equals to 0.154).

  Regression-based imputation methods show slightly better cluster recovery (ARI equals to 0.176) in *Wk*-Means for the Heart dataset. However, in an intelligent variant, all the imputation methods have same effect.

  Wk-Means shows better cluster recovery when missing values are replaced by the feature average or by simple regression for the Iris dataset. In case of *iWk*-Means, cluster-based regression method has better cluster recovery with ARI equal to

0.631 closely followed by KNN imputation (ARI equals to 0.63), simple regression (ARI equals to 0.619) and the feature average, ARI equals to 0.618.

The Wine dataset has slightly better cluster recovery from *Wk*-Means when simple regression is used as imputation methods. For intelligent variant of *Wk*-Means, regression-based imputation methods shows better cluster recovery.

Finally, in the Zoo dataset, cluster-based regression results in better cluster recovery for *Wk*-Means with ARI equals to 0.593. *iWk*-Means has better cluster recovery for the dataset when feature average and KNN, used as imputation methods. Overall, cluster-based regression imputation method results better cluster recovery in both *Wk*-Means and *iWk*-Means with best ARIs in four out of seven and five out of seven respectively.

- Percentages Missing
  In the given configuration, it is hard to find the pattern of cluster recovery for both *Wk*-Means and *iWk*-Means with respect to the percentage of missing values. Feature average and simple regression shows better cluster recovery when percentage of missing values is increased in *iWk*-Means.

- Noise Type
  Simple regression yields better cluster recovery for *Wk*-Means with average ARI equal to 0.472 and is closely followed by cluster-based regression (average ARI equal to 0.472), feature average (average ARI equal to 0.469) and KNN (average ARI equal to 0.467). In intelligent variant, replacing the missing values with feature average perform better with ARI equal to 0.493 followed by cluster-based regression, simple regression and KNN with average ARIs equal to 0.49, 0.489 and 0.48 respectively.

  With addition of noise, cluster-based regression improved the performance of *Wk*-Means; ARIs value for half noise is equal to 0.485 and the average ARI value for full noise is 0.463. Moreover, addition of noisy features up to 50% ; cardinality of the feature space improve the performance of *Wk*-Means in all imputation methods. In *iWk*-Means, replacing the missing values with feature average has better cluster recovery although ARI dropped from 0.493 to 0.424 when full noise is added to the dataset. Addition of noise has similar effects in other imputation methods as well for *iWk*-Means.

B.ii) **Univariate missing pattern with maximum correlation**
In table 6.4, four different imputation methods - feature average, kNN, simple regression and cluster-based regression are observed based on their effect on the performance of *Wk*-Means and Intelligent Weighted *k*-Means on real-world data. For the case study, missing values are present in the feature with maximum correlation with its dataset.

- datasets
  In the Ecoli dataset, KNN imputation results in better cluster recovery for *Wk*-Means and is closely followed by cluster-based regression and then feature average. For *iWk*-Means, all four imputation methods have similar effect with zero deviation (standard deviation equal to 0).

TABLE 6.4: Comparison of four different imputation methods in the real-world (UCI) datasets where missing values have MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

MAXIMUM CORRELATION	*Wk*-Means				Intelligent *Wk*-Means			
	F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
Dataset	Seven different real world databases							
Ecoli	0.146/0.17	**0.244/0.2**	0.19/0.17	0.23/0.19	**0.038/0**	**0.038/0**	**0.038/0**	**0.038/0**
Glass	0.163/0.05	**0.165/0.05**	0.163/0.05	0.164/0.05	**0.076/0.06**	0.049/0.05	0.039/0.04	0.031/0.04
Heart	**0.182/0.11**	0.18/0.11	0.174/0.11	0.174/0.11	0.169/0.02	0.167/0.03	**0.176/0**	**0.176/0**
Iris	0.695/0.13	0.732/0.15	**0.743/0.16**	0.737/0.15	0.589/0.03	**0.625/0.02**	0.618/0.02	0.618/0.01
Soya	0.62/0.18	**0.647/0.19**	0.639/0.21	0.627/0.19	0.927/0.1	0.863/0.14	**0.94/0.08**	0.914/0.1
Wine	**0.795/0.1**	0.778/0.09	0.781/0.07	0.775/0.08	0.58/0.05	**0.618/0.04**	0.595/0.05	0.603/0.04
Zoo	0.573/0.15	**0.579/0.16**	0.561/0.16	0.564/0.16	0.768/0.06	0.79/0.08	0.828/0.09	**0.885/0.06**
Missing %	Evaluation based on % of missing values							
1	0.472/0.3	**0.473/0.3**	0.467/0.3	0.466/0.3	0.457/0.35	0.459/0.35	**0.471/0.36**	0.465/0.36
10	0.45/0.31	**0.463/0.3**	0.459/0.3	0.462/0.29	0.438/0.33	0.457/0.35	**0.472/0.36**	0.463/0.36
15	0.465/0.29	**0.492/0.29**	0.48/0.3	0.479/0.3	0.451/0.33	0.449/0.33	0.456/0.35	**0.471/0.37**
20	0.431/0.29	**0.462/0.29**	0.452/0.3	0.456/0.31	0.451/0.33	0.441/0.32	0.462/0.35	**0.465/0.36**
25	0.452/0.28	**0.483/0.28**	0.466/0.29	0.468/0.28	0.454/0.33	0.451/0.33	0.46/0.35	**0.47/0.36**
30	0.458/0.29	**0.481/0.28**	0.474/0.29	0.474/0.28	0.451/0.33	0.439/0.32	0.459/0.35	**0.466/0.36**
35	0.446/0.3	**0.471/0.28**	0.453/0.29	0.467/0.27	0.444/0.33	0.453/0.34	0.454/0.34	**0.464/0.35**
Noise Type	Performance of the imputation methods with added uniform noisy features							
No Noise	0.453/0.29	**0.475/0.29**	0.464/0.3	0.467/0.29	0.449/0.33	0.45/0.34	0.462/0.35	**0.466/0.36**
Half Noise	0.462/0.25	**0.476/0.25**	0.475/0.25	0.474/0.25	0.444/0.33	0.447/0.33	0.455/0.33	**0.457/0.33**
Full Noise	0.435/0.25	0.447/0.25	**0.45/0.25**	0.444/0.24	0.395/0.29	0.393/0.3	**0.409/0.3**	0.405/0.31

Replacing the missing values with KNN provides the best result for *Wk*-Means in the Glass dataset with average ARI = 0.165. The next best is cluster-based regression, followed by simple regression and feature average. For *iWk*-Means, feature average results in the best cluster recovery, followed by KNN, simple regression and cluster-based regression respectively.

In the Heart dataset, feature average is the first choice imputation method with ARI equal to 0.182 for *Wk*-Means, while regression-based imputation has the worst effect for *Wk*-Means. In contrast, *iWk*-Means has the best cluster recovery for regression based imputation with ARIs equal to 0.176 with absolutely zero deviation for the dataset.

The Iris dataset shows better cluster recovery for *Wk*-Means with simple regression and KNN for *iWk*-Means. In the former, the second best is cluster-based regression followed by KNN and feature average. Later, the second best is the regression based imputation method followed by feature average imputation.

KNN is the best imputation method in the Soya dataset for *Wk*-Means, with average ARI value to 0.647. For intelligent *Wk*-Means, replacing the missing values with simple regression results for the best alternative.

Replacing the missing values with the feature average provides the best cluster recovery for *Wk*-Means in the Wine dataset. The dataset has best cluster recovery when using intelligent *Wk*-Means for KNN imputation.

In the Zoo dataset, optimal cluster recovery for *Wk*-Means is obtained with KNN whereas, for *iWk*-Means cluster-based regression gives the best cluster recovery.

Overall, KNN is the best imputation methods for *Wk*-Means in the real-world

(UCI) dataset. However, a similar conclusion cannot be made for *iWk*-Means as KNN, simple regression and cluster-based regression show better results in three, and feature average in two cases out of seven.

- Percentage Missing
  We cannot observe any trend in cluster recovery for *Wk*-Means when the impact of the four imputation methods is categorized based on the percentage of missing values. However, in *iWk*-Means, when the missing values are less than or equal to 10%, simple regression results in the best cluster recovery and cluster-based regression is found to be best as the percentage of missing values is increased above 10.

- Noise Type
  With cardinality of noisy features less than or equal to half of the feature space cardinality, KNN imputation results in better cluster recovery for *Wk*-Means and cluster-based regression results in better cluster recovery for *iWk*-Means. Simple regression is the best imputation method for both *Wk*-Means and intelligent *Wk*-Means when the number of noisy feature is increased by more than the half of the cardinality of the feature space.

Missing values with NMAR Mechanism

In this section, not missing at random(NMAR) mechanism is observed in the synthetic and the real-world datasets.

A) **Synthetic datasets**

For two cases of univariate missing pattern i.e. feature with maximum and minimum correlation, experimental results from four Gaussian mixed model configurations with NMAR mechanism are explained in this section.

A.i) **Univariate missing pattern with minimum correlation**

In table 6.5, ARI values for *Wk*-Means and *iWk*-Means are listed from the synthetic datasets where missing values have NMAR mechanism and missing values in a feature with minimum correlation with its dataset. Results are categorized based on datasets, percentage of missing values and noise type.

- Dataset Configuration
  Four imputation methods have some effect on the performance of *Wk*-Means with low feature space cardinality ($|V| = 8$). However, replacing the missing values with feature average provides better cluster recovery in low feature space cardinality for *iWk*-Means . The next best case in *iWk*-Means is observed in KNN followed by cluster-based regression and simple regression.

  All imputation methods have the same effect on the performance of *Wk*-Means in datasets with larger feature space cardinality ($|V| = 20$). However, KNN is found to be the best imputation when cardinality of dataset is relatively higher.

TABLE 6.5: Comparison of four different imputation methods in Gaussian mixed models where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has the lowest correlation with its dataset.

MINIMUM CORRELATION		*Wk*-Means				Intelligent *Wk*-Means			
		F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
	DataSet	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5							
	1000x8	**0.971/0.04**	**0.971/0.04**	**0.971/0.04**	**0.971/0.04**	**0.206/0.06**	0.202/0.06	0.198/0.06	0.199/0.06
	1000x12	**0.921/0.18**	0.92/0.18	**0.921/0.18**	**0.921/0.18**	**0.381/0.13**	**0.381/0.13**	0.376/0.13	0.379/0.13
	1000x16	0.881/0.17	0.881/0.17	**0.882/0.17**	0.881/0.17	**0.626/0.12**	0.625/0.12	0.622/0.12	0.622/0.12
	1000x20	**0.857/0.15**	**0.857/0.15**	**0.857/0.15**	**0.857/0.15**	0.848/0.1	**0.849/0.09**	0.847/0.09	0.848/0.09
	Missing %	Evaluation based on % missing values							
	1	**0.909/0.15**	**0.909/0.15**	**0.909/0.15**	**0.909/0.15**	**0.513/0.27**	**0.513/0.27**	**0.513/0.27**	**0.513/0.27**
	10	**0.906/0.15**	**0.906/0.16**	**0.906/0.16**	**0.906/0.16**	**0.512/0.27**	0.511/0.27	0.51/0.27	**0.512/0.27**
	15	**0.908/0.15**	**0.908/0.15**	**0.908/0.15**	**0.908/0.15**	**0.51/0.27**	**0.51/0.27**	0.509/0.27	**0.51/0.27**
	20	0.906/0.15	0.906/0.15	**0.907/0.15**	0.906/0.15	0.509/0.27	**0.51/0.27**	0.509/0.27	**0.51/0.27**
	25	**0.906/0.15**	0.905/0.16	**0.906/0.15**	0.905/0.16	**0.51/0.27**	**0.51/0.27**	0.509/0.27	**0.51/0.27**
	30	**0.91/0.15**	0.909/0.15	**0.91/0.15**	**0.91/0.15**	**0.517/0.26**	0.515/0.26	0.509/0.26	0.511/0.27
	35	0.906/0.15	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	**0.536/0.25**	0.528/0.25	0.516/0.26	0.517/0.26
	Noise Type	Performance of the imputation methods with added uniform noisy features							
	No Noise	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	**0.515/0.26**	0.514/0.27	0.511/0.27	0.512/0.27
	Half Noise	0.853/0.23	**0.854/0.23**	**0.854/0.23**	**0.854/0.23**	**0.797/0.23**	**0.797/0.23**	**0.797/0.23**	**0.797/0.23**
	Full Noise	0.818/0.21	0.818/0.21	0.819/0.21	**0.82/0.21**	0.823/0.18	**0.824/0.18**	**0.824/0.18**	**0.824/0.18**

Wk-Means's ARIs keep decreasing when cardinality of feature space is increased in all four imputation methods but with the opposite scenario in *iWk*-Means. ARIs for *iWk*-Means is increased in all imputations with the increase of feature space cardinality.

- Percentage Missing
  Feature average imputation has best result with 30% missing values (average ARI equals 0.91) and the worst when the missing percentages are 35%, 25% and 20% respectively for *Wk*-Means. *Wk*-Means has the best cluster recovery at 1% and worst cluster recovery at 25% of missing values when KNN is used as an imputation. Simple regression imputation offers the best cluster recovery for *Wk*-Means when the dataset has 30% of missing values, and the worst cluster recovery at 25% and 10% of missing values. It is likely, cluster-based regression results the best option for *Wk*-Means when there are 30% and worst for 25% of missing values. This shows that none of the imputation methods follow a trend with the number of missing values for *Wk*-Means.

  In the case of *iWk*-Means, all of the imputation methods have a similar effect when the percentage of missing values is relatively low (around 1%). Simple regression is outperformed by the other methods with the increment of missing values. When the percentage of missing value is around or above 30%, replacing the missing values with feature average results in better cluster recovery for *iWk*-Means.

- Noise Type
  When there is no noise in the dataset, all four imputation methods have a similar effect on the performance of *Wk*-Means. The average ARI reading of *Wk*-Means for all of the imputation methods is 0.907, with standard deviation equal to 0.15. For *iWk*-Means, replacing the missing values with feature average outperform the other imputation methods.

Feature average is not the best option for either *Wk*-Means or intelligent *Wk*-Means when datasets have noisy features added. Cluster-based regression is the best imputation method for *Wk*-Means with full noise type whereas all three imputation methods except feature average can be chosen for cluster recovery in *iWk*-Means.

A.ii) **Univariate missing pattern with maximum correlation**
In table 6.6, ARI values for *Wk*-Means and *iWk*-Means are listed from the synthetic datasets where missing values have NMAR mechanism and missing values in a feature with maximum correlation with its dataset. Results are categorized based on datasets, percentage of missing values and noise type.

TABLE 6.6: Comparison of four different imputation methods in Gaussian mixed models where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

MAXIMUM CORRELATION	*Wk*-Means				Intelligent *Wk*-Means			
	F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
Dataset	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5							
1000x8	0.944/0.07	0.943/0.07	0.947/0.07	**0.948/0.07**	0.191/0.07	**0.239/0.08**	0.193/0.05	0.208/0.06
1000x12	0.917/0.18	0.913/0.18	**0.918/0.18**	0.913/0.18	0.361/0.11	**0.413/0.11**	0.361/0.1	0.388/0.11
1000x16	**0.882/0.17**	0.877/0.17	0.88/0.17	0.878/0.17	0.591/0.11	**0.633/0.11**	0.605/0.11	0.628/0.11
1000x20	**0.856/0.15**	0.853/0.15	0.854/0.15	0.854/0.15	0.837/0.09	0.832/0.09	0.832/0.09	**0.839/0.09**
Missing %	Evaluation based on % of missing values							
1	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	**0.907/0.15**	0.504/0.26	**0.515/0.27**	0.513/0.27	**0.515/0.27**
10	**0.907/0.15**	0.906/0.15	0.906/0.15	0.906/0.15	0.487/0.27	0.515/0.26	0.5/0.26	**0.517/0.26**
15	**0.898/0.16**	**0.898/0.16**	**0.898/0.16**	**0.898/0.16**	0.484/0.27	**0.52/0.25**	0.498/0.26	0.515/0.26
20	**0.9/0.15**	0.898/0.15	**0.9/0.15**	0.899/0.15	0.483/0.27	**0.527/0.24**	0.494/0.26	0.516/0.26
25	**0.897/0.15**	0.893/0.16	0.896/0.15	0.895/0.16	0.488/0.26	**0.535/0.23**	0.494/0.26	0.517/0.26
30	**0.897/0.15**	0.89/0.15	0.896/0.15	0.893/0.16	0.5/0.25	**0.544/0.23**	0.492/0.25	0.516/0.25
35	0.892/0.15	0.883/0.16	**0.894/0.15**	0.889/0.16	0.517/0.25	**0.55/0.22**	0.493/0.25	0.515/0.25
Noise Type	Performance of the imputation methods with added uniform noisy features							
No Noise	**0.9/0.15**	0.897/0.15	**0.9/0.15**	0.898/0.15	0.495/0.26	**0.529/0.24**	0.498/0.26	0.516/0.26
Half Noise	0.834/0.25	0.84/0.24	**0.843/0.24**	**0.843/0.24**	0.768/0.26	0.784/0.24	0.783/0.24	**0.793/0.24**
Full Noise	0.797/0.22	0.803/0.22	**0.811/0.21**	**0.811/0.21**	0.793/0.21	0.804/0.2	0.809/0.19	**0.816/0.19**

- Dataset Configuration
  Replacing the missing values with cluster-based regression methods results in better cluster recovery for *Wk*-Means when the cardinality of feature space is relatively low ($|V| = 8$). When the cardinality of the feature space is 12, simple regression outperformed the other three imputation methods. Feature average is the best method to replace the missing value for *Wk*-Means when a Gaussian dataset has higher feature space cardinality, M $\geq$ 16.

  For *iWk*-Means, KNN imputation is the best method for replacing the missing values when the feature space cardinality, $|V| \leq 16$. For the datasets with higher feature space cardinality($|V| = 20$), cluster-based regression method outperformed other imputation methods.

- Percentage Missing
  When the percentage of missing values is relatively low (around 1%), all four imputation methods have similar effects on the performance of *Wk*-Means and *iWk*-Means. All imputation methods show the same effect on the performance of *Wk*-Means and *iWk*-Means when the percentage of missing values is equal to or below 15 except that simple regression worsens the performance of *iWk*-Means

when the percentage of missing values is higher than 1%.

In case of a relatively high percentage of missing value, all imputation methods have a similar effect for *Wk*-Means except for feature average, which decreases the performance of *Wk*-Means. The feature average imputation method results in better cluster recovery with mean ARI equals to 0.536 and the worst case is observed in simple regression with mean ARI equals to 0.517 for *iWk*-Means with a higher percentage of missing values .

- Noise Type
  All four imputation methods have a similar effect on the performance of *Wk*-Means when datasets is free of noise. However, for *iWk*-Means, when there is no noise in the dataset, feature average is the best imputation methods followed by KNN, cluster-based regression and simple regression.

  Feature average imputation methods are outperformed by other imputation methods in *Wk*-Means with addition of noisy features. When the number of noisy features is equal to the cardinality of feature space (full noise), cluster-based regression is the best method for *Wk*-Means. In the same scenario, for *iWk*-Means all imputation methods have similar effects except feature average which is the worst choice for the given condition.

A) **Real-World datasets**

In this section, results from two cases of univariate missing pattern with NMAR mechanism in seven real-world datasets are explained.

A.i) **Univariate missing pattern with minimum correlation**

In table 6.7 below, four different imputation methods- feature average, KNN, simple regression and cluster-based regression are observed, based on their effect on the performance of *Wk*-Means and intelligent weighted *k*-Means applied to real-world datasets. For the case study, missing values are present in the feature with minimum correlation with its dataset.

TABLE 6.7: Comparison of four different imputation methods in the real-world datasets where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset.

MINIMUM CORRELATION		WKMeans				Intelligent WKMeans			
		F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
	DataSet	Seven different real world database							
	Ecoli	**0.196/0.17**	**0.196/0.17**	**0.196/0.17**	**0.196/0.17**	**0.038/0**	**0.038/0**	**0.038/0**	**0.038/0**
	Glass	**0.203/0.04**	0.202/0.04	**0.203/0.04**	0.2/0.04	0.155/0.08	0.148/0.09	**0.157/0.09**	0.121/0.09
	Heart	0.186/0.11	**0.187/0.11**	**0.187/0.11**	**0.187/0.11**	**0.176/0**	**0.176/0**	**0.176/0**	0.175/0
	Iris	0.772/0.13	**0.774/0.13**	0.764/0.13	0.768/0.14	0.621/0.01	**0.626/0.02**	0.611/0.02	0.618/0.02
	Soya	0.65/0.19	0.653/0.19	0.644/0.21	**0.676/0.21**	0.84/0.16	**0.859/0.15**	0.855/0.14	0.845/0.14
	Wine	0.78/0.09	0.787/0.09	0.788/0.09	**0.791/0.08**	**0.642/0.03**	0.633/0.03	**0.642/0.02**	**0.642/0.02**
	Zoo	0.611/0.17	**0.616/0.16**	0.606/0.17	0.604/0.17	**0.906/0**	**0.906/0**	**0.906/0**	**0.906/0**
	Missing %	Evaluation based on % missing values							
	1	0.464/0.28	0.463/0.28	0.473/0.29	**0.481/0.29**	0.457/0.35	0.459/0.35	0.46/0.35	**0.461/0.35**
	10	0.507/0.3	**0.508/0.3**	0.498/0.29	0.499/0.3	0.476/0.33	0.476/0.34	**0.485/0.34**	0.465/0.36
	15	0.479/0.29	0.495/0.29	0.499/0.3	**0.504/0.3**	0.483/0.34	**0.491/0.34**	0.483/0.34	0.468/0.34
	20	0.482/0.29	0.48/0.29	**0.483/0.29**	0.48/0.3	**0.488/0.34**	0.486/0.34	0.484/0.34	0.475/0.34
	25	0.501/0.3	**0.505/0.29**	0.495/0.29	0.498/0.3	**0.488/0.33**	0.483/0.33	0.487/0.33	0.483/0.33
	30	0.479/0.3	**0.491/0.31**	0.473/0.3	0.489/0.3	0.494/0.33	**0.497/0.33**	**0.497/0.33**	**0.497/0.33**
	35	**0.487/0.31**	0.473/0.3	0.465/0.3	0.474/0.3	0.491/0.33	0.492/0.33	0.488/0.32	**0.496/0.33**
	Noise Type	Performance of the imputation methods with added uniform noisy features							
	No Noise	0.486/0.29	0.488/0.29	0.484/0.29	**0.489/0.3**	0.483/0.33	**0.484/0.34**	**0.484/0.34**	0.478/0.34
	Half Noise	0.482/0.25	0.482/0.25	0.483/0.25	**0.488/0.26**	0.458/0.34	0.461/0.34	0.46/0.34	**0.464/0.34**
	Full Noise	**0.469/0.26**	0.465/0.26	0.462/0.25	0.466/0.25	0.395/0.31	0.395/0.31	0.394/0.31	**0.397/0.31**

- Dataset
  All four imputation methods show no effect on the performance of *Wk*-Means and *iWk*-Means in Ecoli dataset. The best imputation method for the Glass dataset are feature average and simple regression for *Wk*-Means and simple regression for intelligent *Wk*-Means.

  In the Heart dataset, all four imputation methods except feature average for *Wk*-Means and cluster-based regression for intelligent *Wk*-Means are good options to replace the missing values. KNN is the best in the Iris dataset for both *Wk*-Means and *iWk*-Means whereas the next best for both algorithms is feature average, followed by cluster-based regression.

  Cluster-based regression is the best imputation method for *Wk*-Means and KNN is the best for *iWk*-Means in the Soya dataset. The order (descending) of other imputation based on performance of *Wk*-Means is KNN, feature average and simple regression. Similarly, the order of other imputation methods for intelligent *Wk*-Means is simple regression cluster-based regression and feature average.

  Wk-Means provides the best cluster recovery in the Wine dataset when the missing values are replaced by cluster-based regression. However, for *iWk*-Means, all three methods of imputation: KNN, simple regression and cluster-based regression are equally better than feature average.

  Feature average is found to be the best imputation method for *Wk*-Means in the Zoo dataset, closely followed by cluster-based regression, KNN and simple regression. Furthermore, the cluster-based regression method is the best imputation for *iWk*-Means, followed by KNN, feature average and simple regression, in descending order.

- Percentage Missing

In the table above, with 1% missing values the cluster-based regression method provides better cluster recovery for *Wk*-Means and *iWk*-Means with average ARI equal to 0.481 and 0.461 respectively. In case of 30% missing values, KNN is the best imputation method for *Wk*-Means and KNN, whereas simple regression and cluster-based regression are all better options for *iWk*-Means. And finally feature average and cluster-based regression are best imputation methods for *Wk*-Means and *iWk*-Means. Cluster-based regression and feature average are found to be the best imputation methods in *Wk*-Means for relatively low and high missing values. However, no such pattern is observed in *iWk*-Means.

- Noise Type
  Cluster-based regression is found to be the best imputation method when the number of noisy features are below the half of the cardinality of the feature space for *Wk*-Means. When the number of noisy features to the cardinality of feature space feature average is the best imputation method for *Wk*-Means. In case of *iWk*-Means, KNN and simple regression are the best imputation methods when there is no noisy feature and with the addition of noisy features the cluster-based regression is found to be best imputation method.

A.ii) **Univariate missing pattern with maximum correlation**

In table 6.8 below, four different imputation methods:- feature average, kNN, simple regression and cluster-based regression are observed based on their effect on the performance of *Wk*-Means and Intelligent Weighted *k*-Means on real-world data. For the case study, missing values are present in the feature with minimum correlation with its dataset.

TABLE 6.8: Comparison of four different imputation methods in the real-world datasets where missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

		Wk-Means				Intelligent *Wk*-Means			
		F Avg	KNN	S Regression	C Regression	F Avg	KNN	S Regression	C Regression
MAXIMUM CORRELATION	Dataset	Seven different real world databases							
	Ecoli	0.155/0.19	0.258/0.19	0.247/0.2	**0.299/0.18**	**0.038/0**	**0.038/0**	**0.038/0**	**0.038/0**
	Glass	0.153/0.04	0.154/0.04	**0.16/0.04**	0.159/0.04	0.064/0.02	**0.07/0.03**	0.048/0.04	0.058/0.03
	Heart	0.181/0.1	0.183/0.1	0.183/0.09	**0.184/0.09**	0.145/0.05	0.15/0.05	**0.174/0**	**0.174/0**
	Iris	0.674/0.15	0.708/0.16	**0.734/0.17**	0.727/0.17	0.57/0.03	**0.62/0.02**	0.612/0.02	0.614/0.02
	Soya	0.631/0.18	**0.635/0.18**	0.631/0.16	0.61/0.15	0.832/0.16	0.837/0.16	0.855/0.14	**0.858/0.15**
	Wine	**0.802/0.06**	0.794/0.07	0.783/0.06	0.786/0.06	0.572/0.04	**0.612/0.04**	0.584/0.05	0.597/0.04
	Zoo	0.573/0.15	0.596/0.16	0.614/0.16	**0.63/0.15**	0.796/0.07	0.791/0.08	**0.884/0.06**	0.837/0.08
	Missing %	Evaluation based on % of missing values							
	1	0.461/0.29	0.464/0.29	**0.483/0.28**	0.468/0.28	0.451/0.34	0.456/0.35	**0.462/0.35**	**0.462/0.36**
	10	0.449/0.28	0.475/0.28	0.47/0.29	**0.486/0.29**	0.435/0.33	0.443/0.34	0.448/0.34	**0.464/0.35**
	15	0.454/0.3	0.491/0.3	**0.495/0.3**	0.489/0.29	0.426/0.33	0.433/0.32	0.457/0.35	**0.461/0.35**
	20	0.456/0.28	0.472/0.28	0.477/0.28	**0.497/0.27**	0.433/0.33	0.44/0.32	**0.461/0.35**	0.456/0.34
	25	0.455/0.29	**0.486/0.29**	0.477/0.29	0.479/0.28	0.416/0.31	0.45/0.34	**0.457/0.34**	0.439/0.32
	30	0.464/0.3	0.481/0.29	0.474/0.29	**0.484/0.27**	0.425/0.32	0.445/0.33	**0.462/0.35**	0.448/0.33
	35	0.433/0.29	0.46/0.27	0.475/0.28	**0.492/0.28**	0.43/0.32	**0.45/0.32**	0.448/0.34	0.447/0.33
	Noise Type	Performance of the imputation methods with added uniform noisy features							
	No Noise	0.453/0.29	0.475/0.29	0.479/0.29	**0.485/0.28**	0.431/0.33	0.445/0.33	**0.456/0.34**	0.454/0.34
	Half Noise	0.463/0.27	0.476/0.26	**0.491/0.26**	0.488/0.26	0.424/0.32	0.444/0.33	**0.445/0.33**	0.442/0.33
	Full Noise	0.421/0.26	0.437/0.26	**0.458/0.25**	0.454/0.25	0.368/0.29	**0.384/0.3**	0.382/0.3	**0.384/0.3**

- datasets
  For *Wk*-Means, replacing the missing values with: cluster-based imputation is found to be best for the Ecoli, Heart and Zoo datasets; simple regression is the

best for the Glass and Iris datasets; KNN for Soya and feature average for the Wine dataset. For *iWk*-Means: the Ecoli dataset has equal effect with all of the imputation; the Heart dataset gives better results with regression-based imputation methods, i.e. simple regression and cluster-based regression; the Glass, Iris and Wine datasets share the best results with KNN imputation; the Zoo dataset prefers simple regression; and the Iris dataset goes for the cluster-based regression method.

Overall, cluster-based regression provides the best cluster recovery in most (three out of seven) datasets for *Wk*-Means, followed by simple regression (two out of seven). Whereas, for *iWk*-Means, simple regression and cluster-based regression provide best results in three out of seven datasets.

- Percentage Missing
  In the table above, with 1% missing values simple regression has better cluster recovery for *Wk*-Means, and regression-based imputation methods such as simple regression and cluster-based regression are best for *iWk*-Means. When there are 10% of missing values in the datasets, cluster-based regression is the best imputation method for *Wk*-Means and *iWk*-Means. When the number of missing values increased to: 15% and 20 %, simple and cluster-based regression; 25%, KNN and simple regression; 30%, cluster-based and simple regression; and 35%, KNN and cluster-based regression are the best imputation methods for *Wk*-Means and *iWk*-Means respectively.

  Overall, when the number of missing values is very low simple regression is the best imputation method for *Wk*-Means and cluster-based regression is the best imputation method for *iWk*-Means. Whereas, in case of a relatively high percentage of missing values, KNN imputation results in better cluster recovery for *iWk*-Means.

- Noise Type
  Cluster-based regression and simple regression methods are best for *Wk*-Means and *iWk*-Means respectively when there are no noisy features in the datasets. With addition of noisy features simple regression is found to be the best imputation method for *Wk*-Means. But *iWk*-Means behaves differently with the number of noisy features added. For *iWk*-Means, with addition of noisy features around half the cardinality of feature space simple regression and when the number of added noisy features tends towards (around) the cardinality of feature space, KNN and cluster-based regression are the best imputation methods.

6.4.2 Evaluation of Partial Distance against Different Methods in *k*-Means and Weighted *k*-Means

As we discussed earlier, in this section, the performance of three approaches: imputation, column-wise deletion and partial distance are observed based on ARIs obtained by *k*-Means and *Wk*-Means. Experiments are carried out with both synthetic and real-world (UCI) datasets, with two missing mechanisms (MCAR and NMCR), and with two cases of univariate missing patterns: missing values in a feature with maximum correlation and missing values in a feature with minimum correlation.

Missing values with MCAR Mechanism

In this section, missing completely at random mechanism is observed in synthetic and the real-world datasets.

A) **Synthetic datasets**
For two cases of univariate missing pattern i.e. feature with maximum and minimum correlation, experimental results from four Gaussian mixed model configurations with MCAR mechanism are explained in this section.

A.i) **Univariate missing pattern with minimum correlation**

In this section, we analysed the performance of three approaches to deal with missing values in a Gaussian mixed model. The performance of these approaches are measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have MCAR mechanism and univariate missing pattern with missing values in a feature with minimum correlation with its dataset. Results are categorized based on datasets, percentage of missing values and noise type.

TABLE 6.9: Comparison of two difference clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset.

		F Avg		F Removal		Partial Distance	
		k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
MINIMUM CORRELATION	DataSet	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5					
	1000x8	0.965/0.05	***0.971/0.04***	0.965/0.05	**0.97/0.04**	0.955/0.09	**0.97/0.04**
	1000x12	**0.923/0.17**	0.919/0.18	**0.922/0.18**	0.921/0.18	0.924/0.16	***0.928/0.17***
	1000x16	**0.886/0.17**	0.879/0.17	**0.886/0.17**	0.877/0.17	***0.897/0.16***	0.889/0.16
	1000x20	**0.867/0.15**	0.86/0.15	**0.866/0.15**	0.859/0.15	***0.874/0.15***	0.867/0.14
	Missing %	evaluation based on % of missing values					
	1	**0.921/0.18**	0.919/0.18	**0.92/0.18**	0.917/0.18	***0.924/0.17***	0.919/0.18
	10	**0.933/0.16**	0.931/0.17	***0.934/0.16***	0.932/0.17	0.928/0.16	**0.933/0.16**
	15	0.928/0.17	**0.929/0.17**	**0.925/0.17**	0.924/0.17	0.932/0.16	***0.936/0.16***
	20	**0.925/0.17**	0.918/0.18	**0.923/0.17**	0.919/0.18	***0.929/0.16***	0.927/0.17
	25	**0.934/0.16**	0.933/0.16	**0.932/0.16**	0.931/0.17	0.922/0.17	***0.936/0.16***
	30	**0.921/0.18**	0.92/0.18	0.923/0.17	**0.924/0.18**	0.922/0.16	***0.932/0.16***
	35	**0.924/0.17**	0.922/0.18	0.923/0.17	**0.925/0.17**	0.914/0.17	***0.931/0.16***
	Noise Type	Performance of the partial distance methods with added uniform noisy features					
	No Noise	**0.926/0.17**	0.925/0.17	**0.926/0.17**	0.925/0.17	0.924/0.16	***0.931/0.16***
	Half Noise	0.833/0.29	**0.888/0.2**	0.829/0.29	**0.886/0.2**	0.838/0.28	***0.889/0.2***
	Full Noise	0.823/0.3	**0.849/0.21**	0.82/0.3	***0.861/0.21***	0.829/0.29	**0.856/0.21**

- Dataset Configuration
  Wk-Means provides better cluster recovery for all three approaches when the cardinality of feature space is low (8). However, with the increase in feature space cardinality, *k*-Means is an better alternative for missing values than its weighted variant in all approaches.

Feature average imputation method is the best for *Wk*-Means with average ARI 0.971 when feature space cardinality is low. With increasing in feature space cardinality, *k*-Means with the partial distance has the best cluster recovery and is closely followed by *Wk*-Means with partial distance.
Overall, the partial distance approach is better for dealing with missing values especially in Gaussian datasets with larger feature space cardinality.

- Percentage Missing
  k-Means has better results than its weighted variant when missing value are replaced with the feature average whatever the percentage of missing values. In the case of removing the feature containing missing values, *k*-Means has better cluster recovery then *Wk*-Means when the percentage of missing values is less than or equal to 25%. The partial distance approach clearly favour *Wk*-Means over *k*-Means when the percentage of missing values is greater than or equal to 25%.

  With 1% missing values, the partial distance approach gives the best performance, i.e. *k*-Means with partial distance has an average ARI 0.924. When the percentage of missing values is increased to 10%, column-wise deletion has the best performance, i.e. *k*-Means with column-wise gives an average ARI 0.934. And when the percentage of missing values $\geq$ 20%, the partial distance approach give the best performance especially for *Wk*-Means.

- Noise
  For all three approaches, *k*-Means provides better cluster recovery when there is no noise, while *Wk*-Means gives better performance in Gaussian clusters with added noisy features. Moreover, when the numbers of noisy features are up to half the cardinality of the feature space, the partial distance approach is the best. But, as the number of added noisy feature increases to the cardinality of the feature space, removing the feature containing the missing values results in the best cluster recovery- *Wk*-Means with feature removal has average ARI equals to 0.861.

A.ii) **Univariate missing pattern with maximum correlation**
In this section, we analyse the performance of three approaches to deal with missing values in the Gaussian mixed model. The performance of these approaches is measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have MCAR mechanism and univariate missing pattern with missing values in a feature with maximum correlation with its dataset.

TABLE 6.10: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

MAXIMUM CORRELATION	F Avg		F Removal		Partial Distance	
	k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
Dataset	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5					
1000x8	0.946/0.07	***0.951/0.06***	0.881/0.12	**0.888/0.13**	0.779/0.29	**0.925/0.1**
1000x12	**0.924/0.17**	0.918/0.18	***0.93/0.14***	0.925/0.15	0.888/0.18	**0.924/0.16**
1000x16	**0.892/0.17**	0.882/0.17	***0.899/0.16***	0.891/0.17	**0.892/0.16**	0.891/0.16
1000x20	**0.867/0.15**	0.86/0.15	***0.876/0.15***	0.869/0.15	***0.876/0.14***	0.866/0.15
Missing %	Evaluation based on % of missing values					
1	**0.91/0.15**	0.907/0.15	**0.899/0.14**	0.895/0.15	***0.912/0.15***	0.907/0.15
10	***0.906/0.15***	0.903/0.15	**0.896/0.14**	0.893/0.15	***0.906/0.15***	0.904/0.15
15	**0.909/0.15**	0.903/0.15	**0.898/0.14**	0.895/0.15	***0.91/0.14***	0.904/0.14
20	***0.906/0.15***	0.903/0.15	**0.896/0.15**	0.893/0.15	0.894/0.16	**0.902/0.15**
25	***0.912/0.14***	0.906/0.15	**0.898/0.14**	0.894/0.15	0.858/0.21	**0.9/0.14**
30	***0.905/0.15***	0.9/0.15	**0.896/0.15**	0.894/0.15	0.796/0.26	**0.898/0.15**
35	***0.903/0.15***	0.898/0.15	**0.892/0.15**	0.888/0.15	0.736/0.28	**0.893/0.15**
Noise Type	Performance of the partial distance method with added uniform noisy features					
No Noise	***0.907/0.15***	0.903/0.15	**0.897/0.15**	0.893/0.15	0.859/0.21	**0.901/0.15**
Half Noise	0.803/0.33	***0.844/0.24***	0.703/0.39	**0.794/0.29**	0.768/0.32	**0.828/0.25**
Full Noise	0.792/0.33	***0.808/0.21***	0.678/0.39	**0.751/0.26**	0.749/0.33	**0.777/0.25**

- Dataset Configuration
  As in the previous case, when the missing values are present in the feature with maximum correlation, the weighted variant of *k*-Means provides better results then *k*-Means with all three approaches under study when the feature space cardinality is low (here, 8). And with the increase in feature space cardinality (here more than 8), *k*-Means provides better cluster recovery than its weighted variant.

  When the feature space cardinality of relatively low (here 8), Feature average imputation is found to be relatively better for *Wk*-Means with average ARI = 0.951. With increasing the feature space cardinality, removing the feature containing the noisy features is found to be the best approach to deal with missing values.

- Percentage Missing
  k-Means is found to provide better performance than its weighted variant when feature average imputation or column-wise deletion of features containing missing values are used to deal with missing values. In case of partial distance, *k*-Means than *Wk*-Means when the percentage of missing values is equal to or less than 15%.

  Again, in the case of MCAR mechanism, when missing values are in a feature with maximum correlation, replacing the distance matrix with the partial distance is a better options when the percentage of missing values are below or equal to 15%. And as the percentage of missing values rises above 15%, replacing the missing values with the feature average is the best option for both *k*-Means and *Wk*-Means.

- Noise
  Despite the number of noisy feature added in the Gaussian datasets, the feature average method performs better than feature removal or partial distance approach. In particular, *k*-Means with the feature average imputation method provides better cluster recovery when the percentage of added noisy features is relatively low (or zero). With the addition of noisy features, *Wk*-Means variant surpasses *k*-Means in performance.

B) **Real-World datasets**

Similar to the synthetic datasets, for two cases of univariate missing patterns discussed earlier, experimental results from seven different real-world datasets from UCI machine learning repository are examined with MCAR mechanism in this section.

B.i) **Univariate missing pattern with minimum correlation**

In this section, we analysed the performance of three approaches for dealing with missing values in seven real-world datasets from the UCI machine learning repository. The performance of these approaches is measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have MCAR mechanism and univariate missing pattern with missing values in a feature with minimum correlation with its dataset. The results are categorized based on dataset configuration, percentage of missing values and error type.

TABLE 6.11: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset.

		F Avg		F Removal		Partial Distance	
		k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
MINIMUM CORRELATION	Dataset	Seven different real world databases					
	Ecoli	**0.414/0.06**	0.175/0.18	0.414/0.06	***0.454/0.11***	**0.414/0.07**	0.245/0.19
	Glass	0.182/0.03	**0.185/0.05**	***0.249/0.02***	0.216/0.04	**0.205/0.04**	0.181/0.05
	Heart	**0.337/0.13**	0.18/0.11	***0.34/0.13***	0.187/0.11	**0.315/0.13**	0.181/0.11
	Iris	0.662/0.12	**0.738/0.17**	0.732/0.06	***0.823/0.09***	0.661/0.1	**0.767/0.12**
	Soya	**0.802/0.2**	0.638/0.18	***0.811/0.19***	0.63/0.19	**0.745/0.2**	0.636/0.18
	Wine	***0.853/0.02***	0.784/0.07	**0.843/0.02**	0.802/0.06	**0.838/0.07**	0.779/0.1
	Zoo	**0.662/0.12**	0.563/0.18	***0.675/0.12***	0.556/0.17	**0.654/0.14**	0.563/0.18
	Missing %	Evaluation based on % of missing values					
	1	**0.493/0.25**	0.419/0.28	***0.516/0.25***	0.474/0.27	**0.496/0.27**	0.417/0.28
	10	**0.508/0.28**	0.432/0.31	***0.523/0.27***	0.479/0.29	**0.495/0.27**	0.43/0.31
	15	**0.514/0.27**	0.418/0.29	***0.537/0.26***	0.494/0.28	**0.507/0.26**	0.439/0.29
	20	**0.507/0.28**	0.431/0.3	***0.52/0.27***	0.473/0.28	**0.493/0.26**	0.438/0.3
	25	**0.516/0.27**	0.438/0.3	***0.532/0.27***	0.476/0.28	**0.501/0.27**	0.443/0.28
	30	**0.516/0.28**	0.429/0.29	***0.536/0.28***	0.479/0.28	**0.505/0.26**	0.457/0.28
	35	**0.525/0.28**	0.431/0.29	***0.545/0.28***	0.476/0.27	**0.502/0.27**	0.452/0.28
	Error Type	Performance of the partial distances with added uniform noisy features					
	No Noise	**0.511/0.27**	0.428/0.29	***0.53/0.27***	0.479/0.28	**0.5/0.26**	0.44/0.29
	Half Noise	0.436/0.29	**0.44/0.26**	0.434/0.28	***0.454/0.25***	0.422/0.28	**0.441/0.25**
	Full Noise	**0.424/0.29**	0.423/0.25	0.422/0.29	***0.45/0.25***	0.396/0.27	**0.428/0.25**

- Dataset
  k-Means performs better than *Wk*-Means in five out of seven datasets when missing values are replaced with the feature average or the feature containing the missing values is removed. In the partial distance approach, *k*-Means performs better in six datasets compare to only one dataset for *Wk*-Means.

  Again, removal of the feature containing the missing values is found to be the best approach in six datasets : the Ecoli and Irish datasets performed best with *Wk*-Means, and the Glass, Heart, Soya and Zoo datasets has performed best with *k*-Means. In contrast, the Wine dataset has best cluster recovery with *k*-Means when missing values are replaced with the feature average.

- Percentage Missing
  When we categorized performance of *k*-Means and its weighted variant, *Wk*-Means, *k*-Means has advantages over *Wk*-Means in all three approaches considered. Of the three different approaches, feature removal is the best option for the real-world (UCI) datasets, especially with *k*-Means.

- Noise
  Based on the number of noisy (white noise) features added in the real-world datasets, *k*-Means performs better than *Wk*-Means with feature average imputation in extreme conditions, i.e. when there are no noisy features or when the number of noisy features added is equal to the cardinality of the dataset. However, in case of feature removal or partial distance approaches, *Wk*-Means method is a better tool for cluster recovery than *k*-Means when there are additional noisy(white noise) features in the dataset along with missing values.

B.ii) **Univariate missing pattern with maximum correlation**

In this section, the performances of three approaches are analysed when dealing with missing values in the seven real-world datasets from the UCI machine learning repository. The performance of these approaches is measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have MCAR mechanism and univariate missing pattern with missing values in a feature with maximum correlation with its dataset.

TABLE 6.12: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

		F Avg		F Removal		Partial Distance	
		k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
MAXIMUM CORRELATION	Dataset	Seven different real world databases					
	Ecoli	**0.396/0.06**	0.133/0.17	**0.401/0.07**	0.12/0.14	***0.58/0.09***	0.127/0.19
	Glass	0.16/0.03	**0.163/0.05**	0.157/0.03	**0.162/0.05**	0.159/0.03	***0.164/0.05***
	Heart	***0.333/0.13***	0.174/0.12	**0.314/0.14**	0.161/0.11	**0.317/0.13**	0.166/0.12
	Iris	0.674/0.08	***0.713/0.13***	0.635/0.09	**0.681/0.12**	0.58/0.1	**0.7/0.08**
	Soya	***0.809/0.2***	0.644/0.17	**0.801/0.2**	0.647/0.17	**0.762/0.2**	0.641/0.17
	Wine	**0.863/0.07**	0.803/0.08	***0.915/0***	0.821/0.08	0.77/0.11	**0.776/0.09**
	Zoo	***0.654/0.12***	0.584/0.16	**0.646/0.12**	0.562/0.16	**0.645/0.13**	0.579/0.17
	Missing %	Evaluation based on % of missing values					
	1	***0.508/0.28***	0.439/0.3	**0.503/0.29**	0.429/0.3	**0.489/0.28**	0.439/0.3
	10	***0.509/0.28***	0.42/0.31	**0.504/0.3**	0.414/0.3	**0.506/0.27**	0.407/0.3
	15	***0.523/0.29***	0.427/0.31	**0.522/0.3**	0.42/0.3	**0.52/0.26**	0.418/0.3
	20	***0.505/0.28***	0.428/0.29	**0.497/0.29**	0.417/0.29	**0.498/0.28**	0.408/0.29
	25	***0.502/0.27***	0.412/0.29	**0.496/0.28**	0.402/0.29	**0.5/0.25**	0.394/0.29
	30	***0.504/0.28***	0.403/0.3	**0.497/0.29**	0.402/0.3	**0.478/0.25**	0.395/0.29
	35	***0.504/0.29***	0.423/0.3	**0.498/0.3**	0.406/0.3	**0.475/0.26**	0.422/0.3
	Noise Type	Performance of the partial distance with added uniform noisy features					
	No Noise	***0.508/0.28***	0.422/0.3	**0.503/0.29**	0.413/0.3	**0.495/0.26**	0.412/0.3
	Half Noise	***0.435/0.29***	0.428/0.26	**0.416/0.3**	0.389/0.27	**0.411/0.27**	0.408/0.26
	Full Noise	***0.423/0.29***	0.407/0.26	**0.405/0.3**	0.372/0.26	0.384/0.26	**0.393/0.26**

- Dataset
  Feature average imputation is better options for *k*-Means then its weighted variate as five out of seven datasets results higher cluster recovery for *k*-Means when missing values are replaced with the feature average. The partial distance approach is better for the Glass datasets. *Wk*-Means with partial distance has best cluster recovery among the three approaches, with average ARI 0.164. Replacement of missing value with feature average gives the best cluster recovery: in the Heart dataset, *k*-Means has average ARI= 0.333; in the Iris dataset, *Wk*-Means has average ARI = 0.713; in the Soya dataset, *k*-Means has average ARI = 0.809; in the Wine dataset, *k*-Means has average ARI = 0.863 and in the Zoo dataset, *k*-Means has average ARI = 0.654.

- Percentage Missing In all cases of percentage missing, *k*-Means have better cluster recovery than *Wk*-Means. Replacing the missing values with feature average provides the best results of the three approaches considered, with any number of missing values.

- Noise Type
  The effect of noise favours *k*-Means against *Wk*-Means for all three approaches, except in the single case where partial distance is when the number of added noisy features is equal to the cardinality of the feature space.

Missing values with NMAR Mechanism

In this section, NMAR mechanism is observed in synthetic and real-world datasets.

A) **Synthetic Gaussian Clusters**

For two cases of univariate missing pattern i.e. feature with minimum and maximum correlation, experimental results from four Gaussian mixed model configurations with NMAR mechanism are explained in this section.

A.i) **Univariate Missing Pattern- minimum correlation**

In this section, we analyse the performance of three approaches for dealing with missing values in the Gaussian mixed model. The performances of these approaches are measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have NMAR mechanism and univariate missing pattern with missing values in a feature with minimum correlation with its dataset. The results are categorized based on dataset configurations, percentage of missing values and noise types.

TABLE 6.13: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in Gaussian mixed models. The missing values have a NMAR mechanism and are located in a feature (univariate missing pattern) which has lowest correlation with its dataset.

		F Avg		F Removal		Partial Distance	
		k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
	Dataset	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5					
MINIMUM CORRELATION	1000x8	0.965/0.05	***0.971/0.04***	0.965/0.05	**0.97/0.04**	0.945/0.11	**0.969/0.05**
	1000x12	**0.924/0.17**	0.922/0.18	**0.925/0.17**	0.923/0.18	0.912/0.17	***0.928/0.17***
	1000x16	**0.89/0.17**	0.884/0.17	***0.891/0.17***	0.883/0.17	0.883/0.16	***0.891/0.16***
	1000x20	***0.863/0.15***	0.854/0.15	**0.862/0.15**	0.854/0.15	0.859/0.14	**0.862/0.15**
	Missing %	Evaluation based on % of missing values					
	1	***0.929/0.17***	0.928/0.17	***0.929/0.17***	0.924/0.17	0.922/0.17	**0.926/0.17**
	10	0.924/0.17	**0.926/0.17**	**0.925/0.17**	0.923/0.18	***0.928/0.16***	0.926/0.17
	15	**0.923/0.17**	0.922/0.18	**0.924/0.17**	0.922/0.18	0.923/0.17	***0.927/0.17***
	20	**0.922/0.17**	0.92/0.18	0.923/0.17	**0.926/0.17**	***0.93/0.16***	0.927/0.17
	25	**0.926/0.17**	0.92/0.18	0.925/0.17	**0.926/0.17**	0.916/0.16	***0.934/0.16***
	30	**0.917/0.18**	0.913/0.19	**0.92/0.18**	0.916/0.18	0.903/0.16	***0.923/0.17***
	35	**0.928/0.17**	0.923/0.18	**0.927/0.17**	0.924/0.17	0.86/0.18	***0.935/0.16***
	Noise Type	Performance of the partial distance methods with added uniform noisy features					
	No Noise	**0.924/0.17**	0.922/0.18	**0.925/0.17**	0.923/0.18	0.912/0.17	***0.928/0.17***
	Half Noise	0.832/0.29	***0.885/0.2***	0.829/0.29	**0.884/0.2**	0.821/0.28	***0.885/0.2***
	Full Noise	0.826/0.3	**0.841/0.21**	0.822/0.3	***0.857/0.21***	0.81/0.29	**0.848/0.21**

- Dataset Configuration
  In the case of low feature space cardinality(8), *Wk*-Means provides better cluster recovery than *k*-Means when the missing values are replaced with the feature average or the feature containing the missing values are removed, and vice versa as the feature space cardinality is increased. Replacing the distance matrix with partial distance favours *Wk*-Means over *k*-Means in all datasets.

  Feature average is found to be best among the three approaches in extreme conditions: *Wk*-Means with ARI 0.971 and *k*-Means, with ARI 0.863 for feature space cardinality equals to 8 and 20.

- Percentage Missing
  k-Means performs better than *Wk*-Means with feature average imputation when the percentage of missing values is more than or equal to 15. The feature removal approach favours *k*-Means over *Wk*-Means for large feature space cardinality(greater than or equal to 30%). Furthermore, partial distance approach is better for *Wk*-Means than *k*-Means when the percentages of missing values is more than or equal to 25.

  Feature average (*k*-Means with average ARI 0.929) and feature removal (*k*-Means with average ARI 0.926) are found to be best when the percentage of missing value is relatively low(1%). When the percentage of missing values is increased, the partial distance performs better in *k*-Means with 10% and 20% missing values and in *Wk*-Means with 15%, 25%, 30% and 35% missing values.

- Noise Type

k-Means with the feature average or feature removal approach performs better than its weighted variant when there in no noise in the dataset. However, with the addition of noise, *Wk*-Means with the feature average and feature removal approach performs better than *k*-Means. The Partial distance approach favours *Wk*-Means over *k*-Means for all noise types.

When there is no noise, maximum ARI is observed in *Wk*-Means with partial distance- average ARI equals to 0.928, whereas with full noise, maximum cluster recovery is given by *Wk*-Means with feature removal approach.

A.ii) **Univariate missing pattern with maximum correlation**

In this section, we analyse the performance of three approaches to deal with missing values in the Gaussian mixed model. The performance of these approaches is measured based on ARIs obtained from *k*-Means and *Wk*-Means. The missing values in the experiment have NMAR mechanism and univariate missing pattern with missing values in a feature with maximum correlation with its dataset.

TABLE 6.14: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in a Gaussian mixed model. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has maximum correlation with its dataset.

MAXIMUM CORRELATION	F Avg		F Removal		Partial Distance	
	k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
Dataset	Four different cardinality of Gaussian clusters with covariance matrix equal to 0.5					
1000x8	0.965/0.05	***0.975/0.02***	0.964/0.07	**0.972/0.05**	0.945/0.12	**0.963/0.04**
1000x12	**0.922/0.17**	0.916/0.18	***0.924/0.15***	0.921/0.16	0.812/0.2	**0.922/0.17**
1000x16	**0.893/0.17**	0.885/0.17	***0.9/0.16***	0.894/0.17	0.833/0.17	**0.891/0.17**
1000x20	**0.864/0.15**	0.859/0.15	***0.871/0.15***	0.864/0.15	0.83/0.15	**0.863/0.15**
Missing %	Evaluation based on % of missing values					
1	**0.894/0.17**	0.89/0.17	***0.902/0.15***	0.899/0.16	**0.898/0.17**	0.89/0.17
10	**0.894/0.17**	0.887/0.17	**0.898/0.16**	0.892/0.16	***0.899/0.16***	0.888/0.17
15	**0.89/0.17**	0.883/0.17	***0.901/0.15***	0.895/0.16	0.874/0.16	**0.887/0.16**
20	**0.893/0.17**	0.887/0.17	***0.899/0.15***	0.895/0.16	0.809/0.17	**0.891/0.16**
25	**0.894/0.16**	0.888/0.17	***0.895/0.16***	0.89/0.16	0.763/0.16	**0.894/0.16**
30	**0.893/0.16**	0.888/0.17	***0.895/0.16***	0.892/0.16	0.756/0.17	***0.895/0.16***
35	**0.892/0.17**	0.885/0.17	**0.895/0.16**	0.887/0.16	0.778/0.17	***0.899/0.16***
Noise Type	Performance of the partial distance methods with added uniform noisy features					
No Noise	**0.893/0.17**	0.887/0.17	***0.898/0.16***	0.893/0.16	0.825/0.18	**0.892/0.16**
Half Noise	0.867/0.23	***0.87/0.16***	0.827/0.26	**0.859/0.17**	0.776/0.23	**0.861/0.17**
Full Noise	***0.849/0.24***	0.789/0.18	**0.796/0.28**	0.773/0.18	0.755/0.24	**0.784/0.18**

- Dataset
  k-Means and *Wk*-Means follow a similar pattern to that in previous table 6.13. The feature average, feature removal and partial distance favour *Wk*-Means when the cardinality of feature space is low (8). With increase in number of features, *k*-Means gives better cluster recovery than its weighted variant for feature average imputation and feature removal approach. In case of partial distance, it *Wk*-Means is always better than *k*-Means, for any feature space.

Feature average imputation method is the best: for *Wk*-Means with average ARI 0.975 when the number of features is low (8). With increasing feature space cardinality, feature removal approach (with *k*-Means clustering) is the best among the three approaches.

- Percentage Missing
  k-Means is found to be a better clustering algorithm than its weighted variant for feature average imputation and the feature removal approach irrespective of the percentage of missing values. However, in case of partial distance, *k*-Means performs better than *Wk*-Means where there is a relatively low percentage of missing values (less than or equal to 10%) and vice versa for larger number of missing values (here more than or equal to 15%).

- Noise Type
  k-Means performs better than *Wk*-Means in feature average imputation and the feature removal approach for two extreme cases of noise: no noise and full noise. In case of partial distance, *Wk*-Means has better performance than *k*-Means in all noise type.

  Feature removal approach is the best- *k*-Means with average ARI 0.898 among the three approaches when there is no noise. With the addition of noise, average imputation - *Wk*-Means with average AIR 0.87 in half noise and *k*-Means with average ARI equals to 0.849 in full noise- are the best among three approaches.

B) **Real-world datasets**
In this section, results from two cases of univariate missing pattern with NMAR mechanism in seven real-world datasets are explained.

B.i) **Univariate missing pattern with minimum correlation**

In this section, we analyse the performance of three approaches for dealing with missing values in seven different real-world datasets from the UCI machine learning repository. The missing values in the experiment have NMAR mechanism and univariate missing pattern with missing values in a feature with minimum correlation with its dataset. The results are categorized based on datasets, percentage of missing values and noise type.

TABLE 6.15: Comparison of two difference clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset.

MINIMUM CORRELATION

	F Avg		F Removal		Partial Distance	
	k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
Dataset	Seven different real world databases					
Ecoli	**0.413/0.06**	0.242/0.2	0.413/0.06	***0.447/0.12***	**0.415/0.07**	0.243/0.2
Glass	**0.217/0.04**	0.204/0.04	***0.252/0.02***	0.225/0.04	**0.22/0.04**	0.2/0.05
Heart	***0.358/0.11***	0.178/0.11	**0.355/0.11**	0.184/0.11	**0.333/0.11**	0.178/0.11
Iris	0.68/0.11	**0.757/0.15**	0.723/0.08	***0.821/0.1***	0.604/0.13	**0.733/0.12**
Soya	**0.794/0.21**	0.625/0.18	***0.785/0.21***	0.621/0.18	**0.764/0.2**	0.627/0.18
Wine	***0.853/0.02***	0.784/0.09	**0.842/0.02**	0.79/0.1	0.771/0.13	**0.785/0.11**
Zoo	**0.696/0.13**	0.603/0.17	***0.697/0.13***	0.605/0.17	**0.69/0.14**	0.603/0.17
Missing %	Evaluation based on % of missing values					
1	**0.519/0.27**	0.44/0.28	***0.534/0.26***	0.477/0.27	**0.495/0.27**	0.443/0.28
10	**0.508/0.27**	0.434/0.29	***0.517/0.26***	0.459/0.28	**0.507/0.28**	0.434/0.29
15	**0.528/0.27**	0.447/0.29	***0.539/0.26***	0.49/0.27	**0.513/0.27**	0.447/0.29
20	**0.512/0.26**	0.445/0.28	**0.519/0.26**	0.488/0.27	***0.52/0.27***	0.446/0.28
25	**0.528/0.27**	0.438/0.3	***0.531/0.27***	0.482/0.28	**0.496/0.26**	0.427/0.29
30	**0.531/0.27**	0.447/0.3	***0.535/0.27***	0.485/0.28	**0.469/0.25**	0.448/0.28
35	**0.536/0.28**	0.457/0.3	***0.537/0.28***	0.487/0.28	**0.456/0.22**	0.446/0.29
Noise Type	Performance of the imputation methods with added uniform noisy features					
No Noise	**0.523/0.27**	0.444/0.29	***0.53/0.26***	0.481/0.28	**0.493/0.26**	0.442/0.29
Half Noise	0.442/0.29	**0.452/0.26**	0.438/0.28	***0.468/0.26***	0.41/0.26	**0.448/0.26**
Full Noise	**0.431/0.29**	0.422/0.25	0.428/0.29	***0.446/0.25***	0.389/0.26	**0.421/0.25**

- Dataset
  Imputation of missing value with feature average is the best option for *k*-Means in six datasets- Ecoli, Glass, Heart, Soya, Wine and Zoo. For *Wk*-Means, feature average imputation gives better results only in the Iris dataset. The feature removal approach is found to be better in five datasets: Glass, Heart, Soya, Zoo for *k*-Means. For *Wk*-Means, the feature removal approach gives better cluster recovery in the Ecoli and Iris datasets. Partial distance favour *k*-Means in five datasets- Ecoli, Glass, Heart, Soya and Zoo, whereas *Wk*-Means favours Iris and Wine datasets.

  In comparison with three approaches, Feature removal is found to be best in five datasets- *Wk*-Means in Ecoli and Iris, and *k*-Means in Glass, Soya and Zoo dataset. Whereas feature average imputation is best in two datasets- *k*-Means in Heart and Wine datasets.

- Percentage Missing
  On the basis of percentage of missing values, *k*-Means performs better than *Wk*-Means in all three approaches. The feature removal approach(with *k*-Means) is found to be the best among the three approaches.

- Noise Type
  k-Means with feature removal and partial distance performs better than its weighted variant when there in no noise in the datasets, and vice versa with the addition of noise. In the case of feature average, *k*-Means performs better in extreme cases (no noise and full noise), whereas *Wk*-Means is better when there is half noise in the datasets.

  Among the three approaches, feature removal is best for all type of noise- *k*-Means with average ARI 0.53 for no noise; and *Wk*-Means with average ARI 0.468 and 0.446 for half and full noise respectively.

B.ii) **Univariate missing pattern with maximum correlation**

In this section, we analysed the performance of three approaches dealing with missing values in seven different real-world datasets from the UCI machine learning repository. The missing values in the experiment have NMAR mechanism and univariate missing pattern with missing values in a feature with maximum correlation with its dataset. The results are categorized based on datasets, percentage of missing values and noise type.

TABLE 6.16: Comparison of two different clustering algorithms- *k*-Means and *Wk*-Means along with three different approaches to handle missing attribute values in real-world datasets. The missing values have a MCAR mechanism and are located in a feature (univariate missing pattern) which has minimum correlation with its dataset.

		F Avg		F Removal		Partial Distance	
		k-Means	*Wk*-Means	*k*-Means	*Wk*-Means	*k*-Means	*Wk*-Means
MAXIMUM CORRELATION	Dataset	Seven different real world databases					
	Ecoli	**0.386/0.06**	0.121/0.16	**0.391/0.06**	0.107/0.14	***0.562/0.11***	0.365/0.25
	Glass	0.156/0.02	**0.157/0.05**	**0.155/0.02**	0.152/0.05	0.155/0.03	***0.171/0.06***
	Heart	**0.327/0.13**	0.18/0.11	**0.322/0.13**	0.167/0.11	***0.358/0.09***	0.173/0.11
	Iris	0.616/0.12	**0.659/0.15**	0.607/0.11	**0.644/0.15**	0.56/0.1	***0.677/0.09***
	Soya	***0.768/0.2***	0.598/0.17	**0.761/0.2**	0.599/0.18	**0.764/0.21**	0.597/0.17
	Wine	**0.874/0.02**	0.786/0.09	***0.915/0***	0.813/0.09	0.67/0.15	**0.78/0.1**
	Zoo	***0.7/0.13***	0.615/0.17	**0.695/0.12**	0.594/0.17	**0.628/0.13**	0.617/0.17
	Missing %	Evaluation based on % missing values					
	1	**0.498/0.27**	0.413/0.29	**0.497/0.28**	0.398/0.3	***0.499/0.27***	0.423/0.29
	10	***0.497/0.28***	0.401/0.3	**0.494/0.29**	0.393/0.3	**0.492/0.27**	0.396/0.29
	15	***0.505/0.28***	0.421/0.29	**0.501/0.29**	0.406/0.3	***0.505/0.26***	0.435/0.28
	20	**0.503/0.28**	0.417/0.29	***0.507/0.29***	0.409/0.29	**0.474/0.23**	0.462/0.27
	25	**0.49/0.28**	0.421/0.29	***0.494/0.29***	0.411/0.29	**0.461/0.24**	0.46/0.26
	30	**0.498/0.28**	0.396/0.29	***0.5/0.29***	0.402/0.3	**0.463/0.26**	0.458/0.27
	35	***0.511/0.28***	0.399/0.29	**0.505/0.29**	0.402/0.29	0.445/0.24	**0.463/0.26**
	Noise Type	Performance of the imputation methods with added uniform noisy features					
	No Noise	***0.5/0.28***	0.41/0.29	***0.5/0.29***	0.403/0.3	**0.477/0.26**	0.442/0.28
	Half Noise	***0.429/0.29***	0.419/0.27	**0.417/0.3**	0.386/0.28	0.381/0.26	**0.412/0.26**
	Full Noise	***0.419/0.3***	0.389/0.26	**0.408/0.31**	0.365/0.26	0.365/0.25	**0.391/0.26**

- Dataset
  Replacing the missing values with the feature average results in better cluster recovery in *k*-Means than *Wk*-Means for five datasets: Ecoli, Heart, Soya, Wine and Zoo. *Wk*-Means has better cluster recovery in the Glass and the Iris datasets. In case of removing the feature containing missing values, *Wk*-Means performs better than *k*-Means in the Iris dataset. Whereas using the partial distance approach performs better for: *k*-Means in the Ecoli, Heart, Soya and Zoo datasets, and *Wk*-Means in the Glass, Iris and Wine datasets.

  Comparing the three approaches, partial distance gives better results in four datasets- *Wk*-Means in Glass (average ARI 0.71) and Iris (average ARI 0.677); and *k*-Means in the Ecoli dataset (average ARI 0.562) and Heart (average ARI 0.358). The feature average imputation performs better in two datasets- *k*-Means in Soya (average ARI 0.768) and Zoo (average ARI 0.7), and in on case best results with obtained with the feature removal approach - *k*-Means in Wine dataset (average ARI 0.915).

- Percentage Missing
  k-Means surpasses its weighted variant, *Wk*-Means, in all three approaches for any percentage of missing values except for the partial distance approach with 35% missing values where *Wk*-Means with average ARI 0.463 is found better than *k*-Means with average ARI 0.445.
  Among the three approaches, with 1% of missing values, the partial distance approach is found to better (*k*-Means with average ARI 0.499) and for other, the feature average and feature removal approaches share the best performance between them, with no apparent trend in respect to percentage of missing values.

- Noise Type
  Replacing the missing value with feature average and removing the feature with missing values favour *k*-Means over *Wk*-Means. The partial distance approach is found to be better suited to *k*-Means than *Wk*-Means when there is no noise, whereas with the addition of noisy features the performance is reversed.
  Feature average (with *k*-Means) surpasses other two approaches- feature removal and partial distance - in all three types of noise cases: no noise, half noise and full noise.

Note:
In the above experiment, linear correlation is used to find a feature with maximum and minimum correlation. Finding the correlation using different statistical tool like Pearson correlation or Rank correlation can be an interesting area for further research, especially when dataset have categorical or rank attributes. For example, the Soya dataset have all categorical features.

Here categorical features are converted into binary features. Therefore, this research is stick with linear correlation.

6.5 Conclusion

Three different approaches: imputation, deletion and partial distance are observed thoroughly in *k*-Means type algorithms. Along with Gaussian clusters, experiments are extended to real-world datasets from the UCI machine learning repository, different missing value configurations

and noise types.

For the experiments carried out the choice of particular approach depends on configuration listed above. The experiments, based on different configuration, can be concluded as:

- Imputation - MCAR

  In synthetic datasets, the performance of *Wk*-Means decreases as the cardinality of feature space increases, whereas performance is higher in *iWk*-Means. The same pattern is observed with the increase of the cardinality for noisy features in both cases of univariate missing patterns.

  In the real-world datasets, only *iWk*-Means shows a pattern of reduced performance for all methods whereas no pattern is observed with percentage of missing values in both *Wk*-Means and *iWk*-Means when missing values are in a feature with minimum correlation. In the case of maximum correlation, KNN is found to be a better imputation method for *Wk*-Means and the simple regression method when the missing percentage is less than 10%; cluster-based regression for the rest is found to be better in *iWk*-Means.

- Imputation - NMAR

  Simple regression is the best imputation method for *Wk*-Means for all dataset configurations and percentages of missing values, although cluster-based regression is a better alternative for full noise. In *iWk*-Means, feature average is the best choice when there is no noise, whereas other imputation methods give better results with full noise.

  In case of maximum correlation, simple regression is the best imputation method for *Wk*-Means for all datasets and percentages of missing values. Cluster-based regression is found to be a better imputation method for *Wk*-Means with all noise types. Feature average is the best imputation method for *iWk*-Means with no noise and other imputation methods are better with full noise except feature average.

  In real-world datasets, cluster-based regression is a better choice when there is no or half noise and feature average is best when there is full noise for *Wk*-Means with minimum correlation. With maximum correlation, regression-based imputations are better for *Wk*-Means and *iWk*-Means.In *Wk*-Means, simple regression is better when there is half and full noise and cluster-based regression is an optimal choice when there is no noise. This is reversed for *iWk*-Means.

- Approaches - MCAR

  In all three approaches, feature average imputation is better in synthetic datasets with higher feature space cardinality for both case of correlation. The partial distance approach is found to be the best option. Among the three approaches, partial distance is found to be better at dealing with missing values for both *k*-Means and *Wk*-Means when missing values are in feature with minimum correlation. However, when missing values are in a feature with maximum correlation, partial distance is found to be better with low number of missing values and feature average imputation for higher numbers of missing values.

In real-world datasets, feature removal is the best approach for both *k*-Means and *Wk*-Means with missing values in feature with minimum correlation but feature average imputation is better for both *k*-Means and *Wk*-Means when there are missing values in features with maximum correlation.

- Approaches - NMAR

  In the case of minimum correlation, partial distance is a better approach in the synthetic datasets for *Wk*-Means in all cases and feature average is the best option for *Wk*-Means, especially when there are noisy features or a high number of missing values. But in the case of maximum correlation, the feature removal approach is better for *k*-Means and partial distance is better for *Wk*-Means when there are high number of missing values.

  In the case of maximum correlation, the feature removal approach is better for *k*-Means and *Wk*-Means in real-world datasets with any percentage of missing values. However, feature average imputation is the best for *k*-Means with noisy feature. When missing values are in the feature with maximum correlation, the partial distance approach is best in four datasets- Ecoli, Glass, Heart and Iris. The feature removal approach is found to be the best one for *k*-Means when there are a large number of missing values, whereas partial distance is the best approach for *Wk*-Means.

Overall, for the synthetic datasets, the cluster-based regression method is better in both *Wk*-Means and *iWk*-Means with noise, irrespective of feature correlation. The regression-based imputation is found to be the better alternative in real-world datasets for both *Wk*-Means and *iWk*-Means, despite feature correlation but not conclusive.

Chapter 7

Performance of *k*-Means types in p-norms

The Minkowski weighted *k*-Means (*MWk*-Means) replaces Euclidean distance by the Minkowski metric (L_p-metric) [92] which enables it to capture different shapes of clusters other than spherical. To our knowledge, the utility of L_p-metric in *MWk*-Means has are yet to explore fully as most of the synthetic datasets used in data mining are generated in 2-norm. In this Chapter, the performance of *k*-Means type algorithms, such as - original *k*-Means (referred as *k*-Means), intelligent *k*-Means (*ik*-Means) [161], Minkowski weighted *k*-Means (*MWk*-Means) [92] and intelligent Minkowski weighted *k*-Means (*iMWk*-Means) [152] - are observed in different p-norms. The adjusted Rand index(RAND) [142] is used to evaluate the performance of these algorithms.

The concept of using polar coordinate to simulate p-generalized Gaussian distribution, practised by Kalke et al. [126], is used to transfer a dataset from one norms to another norms. In order to run the experiments, Gaussian clusters in different p-norms are generated using a Gaussian mixed model (GMM) available in MATLAB. The Gaussian mixed models are hyper-spherical clusters defined in 2-norm. With feature space cardinality (m) equals two, i.e. m=2, these clusters will have a circular shape and can be displayed in a 2-D plane where each of the two feature is represented by x-axis and y-axis represents. When the feature space cardinality is extended to three, i.e. m = 3, each of these Gaussian clusters will have a spherical shape. In general, a Gaussian cluster defined over an m-dimensional feature space is represented by an m-dimension hypersphere and each data feature that defines the cluster corresponds to one of the axes of the hyperplane.

7.1 Aims

Aims of this experiment is to examine the performance of *k*-Means algorithms in different L_p-spaces and explore the usefulness of Minkowksi metric. We limit our experiment to the recovery(transformation) of Gaussian clusters, generated by MATLAB gmm model, in different p-norms. This experiment examines the effect of the distance coefficient (p_{dist}) on performance of *k*-Means types algorithm. To achieve this, we first create a model that translates spherical shaped Gaussian clusters, which are generated using Euclidean matrix as similarity measurement in Euclidean space, to regular shaped clusters in L_p space. As discussed earlier, the shape of translated clusters are defined by a similarity measurement based on Minkowski matrix. This is achieved by altering the distance coefficient (p_{dist}) in the similarity measurement. In the later part, the performance of *k*-Means type algorithms are observed with respect to the shape of the Gaussian cluster.

7.2 Validation of Transformation Process

Transformation of data points from p_{old}-norm to p_{new}-norm which introduced in section 4.3 is validated by reverse engineering. According to the principle of reverse engineering, if the transformation process is correct, the original data points are recovered when the data points are transformed back from p_{new}-norm to p_{old}-norm . For the purpose of experiment, a cluster in two dimensions using the GMM is generated in MATLAB. The cluster is defined in 2-norm by default and hence has a circular shape. The cluster is then moved to the different p-norm which re-shapes the cluster. The process is reversed with the transformation, i.e. clusters in different p-norms are reshaped back to 2-norm for reverse engineering. The whole process is visualised by displaying the cluster in a 2D plane.

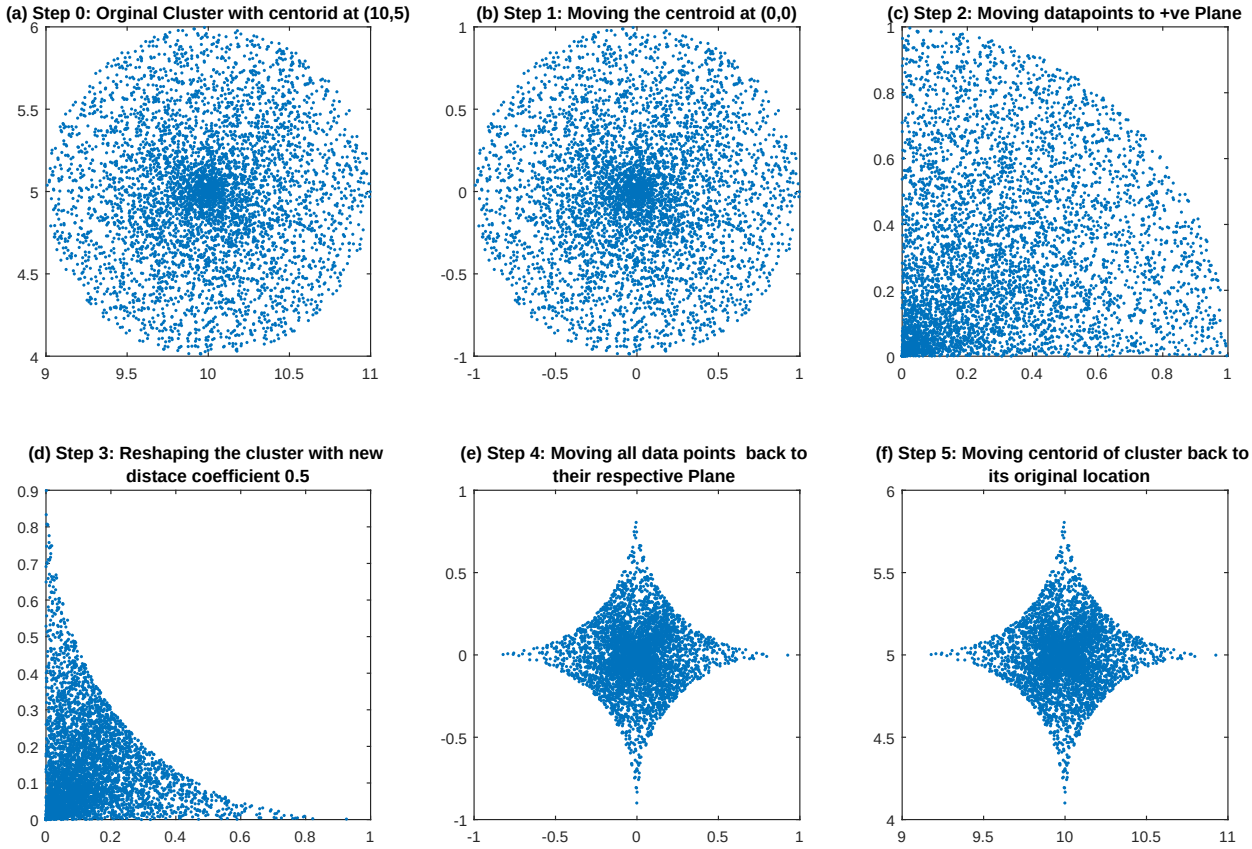


FIGURE 7.1: Transformation of a 2D cluster with center at (10, 5) from 2-norm space to 0.5-norm space. The process is carried out in six steps from top left (a) to bottom right (e).

Visualization of Transformation Process in 2D

A set of data points of size 5000 with center (10, 5) is created using the GMM in Matlab. These data points have a degree ($\theta \in (0, \pi)$ which is drawn from a uniform distribution and a radius $r \in (0, 1)$, also drawn from the uniform distribution. Using the polar to Cartesian coordinate transformation, two coordinates, x and y, are derived for each data point in the cluster.

Based on the behaviour of the transforming process with respect to the value of p, experiments are categorized into two subgroups: p with usual values and p with extreme values. From our experiment, values of p less than 0.05 or greater than 100 are categorized as extreme values.

Categories of p values :

- Case 1, p $\in$ [0.05,1,1.5,2.5,10,100]
- Case 2, p tends to infinity or 0

Case 1: p in the range [0.05,0.5,1,1.5,2.5,4,10,100]

The 2D data points displayed in Figure 7.1.a is first transformed to the corresponding p-norm from 2-norm using the procedure defined in Algorithm 2. The process is then reversed back to 2-norm. This reverse engineering process helps us to check whether the transformation process is valid for p $\in$ [0.05,0.5,1,1.5,2.5,4,10,100].

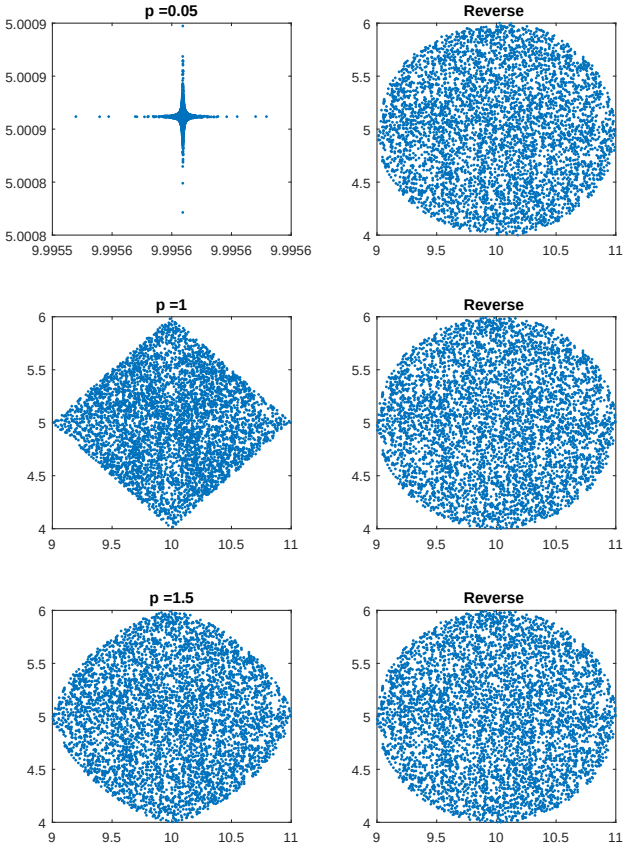


FIGURE 7.2: Results from moving data points, originally defined in 2-norm with center at 10,5), to p-norms ($p \in [0.05, 1, 1.5]$ are displayed in each of the sub-plots on left hand side. Results from the reverse process are displayed sub-plots on the right.

Results from moving data points, originally defined in 2-norm with center at 10,5), to p-norms ($p \in [0.05, 1, 1.5]$ are displayed in each of the sub-plots on left hand side. Results from the reverse process are displayed to the right of each corresponding sub-plot.

The value of p in Figure 7.2 $\in$ {0.05,1,1.5}. Each sub plot in the first column represents the cluster reshaped to the corresponding p-norm. The data points in the right column show the results from reverse engineering where the data points on the left is reshaped back to the 2-norm space.

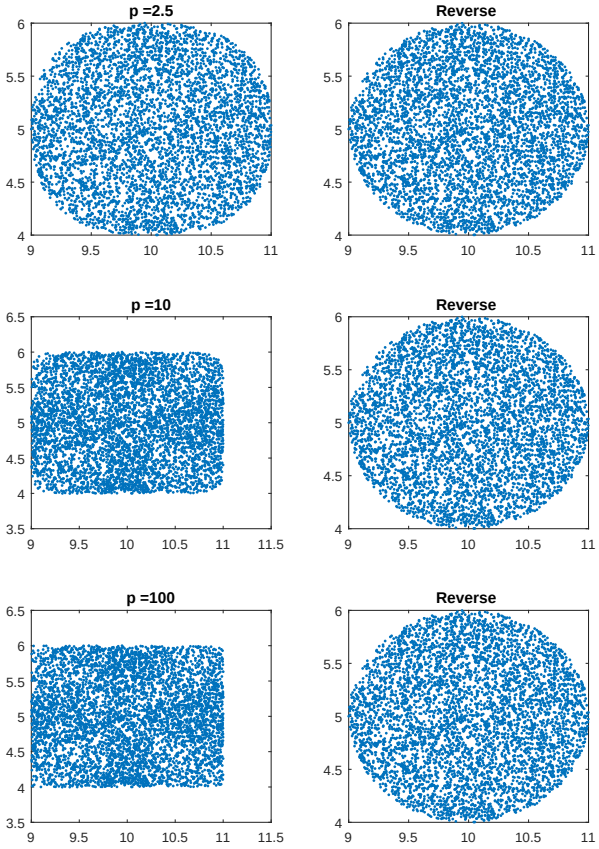


FIGURE 7.3: Data points with center (10,5) defined in two dimensions in Euclidean space is reshaped using a different distance coefficient (p). Each sub plot in the left column represents the data points changed to its respective p-norms. The right column shows the results from reversing the process back to a 2-norm.

The two figures above show the reversal of the process of reshaping the Gaussian clusters. Initially, a Gaussian cluster is generated with the center at (10,5). The data points is then reshaped by transforming the data points in different p-norms. In the second part of the experiment, the process is reversed by transforming the cluster back to 2-norm. The visualization of the process (Figure 7.2 and 7.3 show that the process of reshaping is valid.

Case 2: p moving toward extreme values

When the distance coefficient, p, moves toward the extreme points 0 or ∞- the reshaping is not reversible.

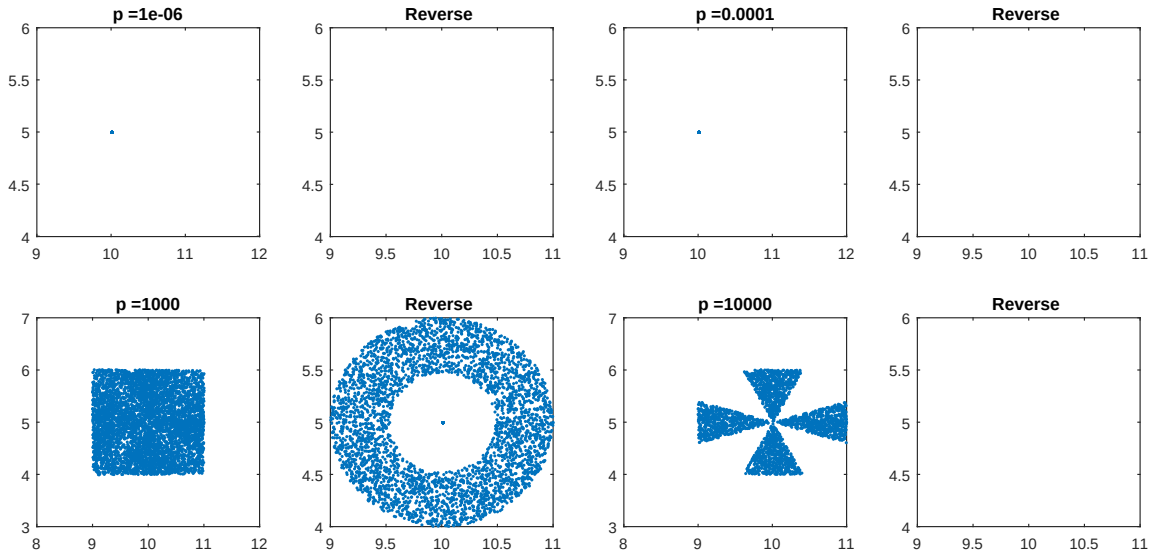


FIGURE 7.4: A set of data points with center (10,5) defined in the two dimensions in Euclidean space is reshaped using some extreme distance coefficient (p). The values of p are either less than 0.005 or greater than 100. To the right of each sub-plot with a given p, a reverse sub plot shows the results obtained when the process is reversed.

As shown in Figure 7.4, when the distance coefficient, p moves toward 0, all points tend to overlap (move close) to the center, and they cannot be reversed. On the other hand, when the values of p tends to infinity, dispersion of data points from their center become larger. All data points tend to move towards ∞ and therefore, may not be reversible. We observed that the reason for not being reversed for extreme values of p is the precision power of the machine (MATLAB). In MATLAB, by default 16 digits of precision are set.

Example:
The precision in MATLAB is 32. When p -> 0, during transformation process some points, let say (a, b) moved to $(2 * 10^{-34}, 5 * 10^{-34})$ becomes (0, 0) and therefore cannot reverse back to (a, b). The similar situation is observed when data points are moved to larger values.

In most publication studies p-norm, usually the distance coefficient lies between 1.1 to 5 [137, 92, 153, 162]. From our experiments, our transformation works well and is reversible as well for the range of distance coefficients well accepted in most of the papers on p-norm (Minkowski distance) [152, 92].

7.2.1 Generation of a Gaussian Mixed Model in Different p-norms

In the Gaussian mixed model, each model is equivalent to a cluster. MATLAB has a built-in Gaussian mixed model library, gmm, that generates a dataset with given K components (clusters) in N by m ($|V|$) dimensions where N represents the total number of data points and m is the number of the features (feature space cardinality).

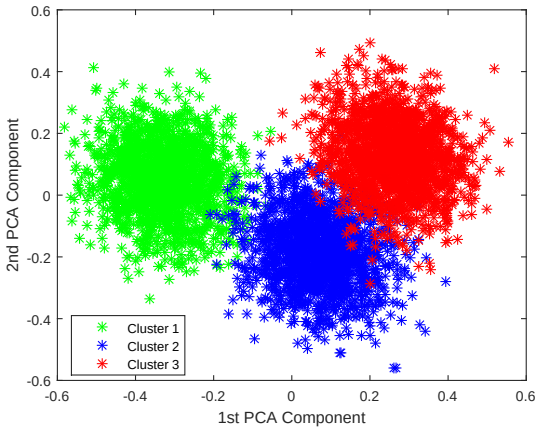


FIGURE 7.5: First and second PCA components view of three Gaussian clusters generated using the MATLAB Gaussian mixed model.

Figure 7.5 displays the first and second principle component analysis (PCA) of three Gaussian clusters, generated using the MATLAB build in module gmm. The dataset has 1000 data points which are equally distributed between three components (clusters) and the feature space cardinality, m = 5. Centroids for each cluster in the Figure 7.5 are generated from a Uniform distribution and the gmm model used has a spherical shape with covariance equal to 0.5.

Translation of GMM into different p-norms

The gmm by default has Cartesian coordiantes and uses distance coefficient equal to 2, i.e. these clusters are defined in 2-norm. Translation of the Gaussian clusters in the Figure 7.5 can be done by translating each Gaussian clusters using the Algorithm 1, where the center of the data points in each cluster is given by the centroid of the respective clusters.

Validation of the transformation for the Gaussian mixed model.

A similar approach of reverse engineering is followed to validate the transformation process for Gaussian mixed model. First, the three Gaussian clusters, in Figure 7.5, are transformed to different p-norms ($p \in [0.5,1,1.5,2.5,5,50]$) and then reversed to 2-norm. Figure 7.6 summarizes the entire process.

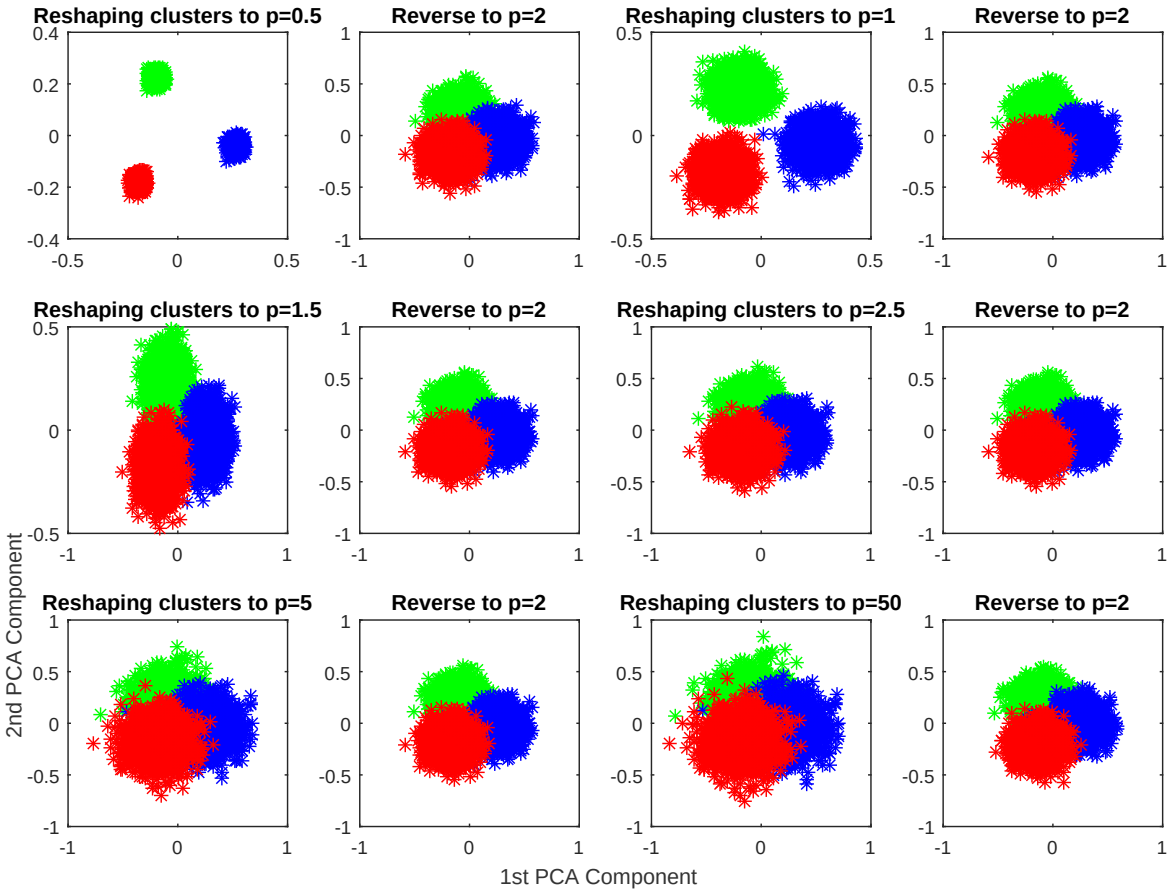


FIGURE 7.6: First and Second PCA component view of 3 Gaussian clusters generated using the MATLAB Gaussian mixed model. Centroids for each of these clusters are generated from a Uniform distribution and the GMM has a spherical shape.

As shown in the figure 7.6, the original clusters at 2-norm are successfully retrieved and hence, the transformation process in GMM is validated.

7.3 Effect of p-norms over Clustering Algorithms

Performance of four *k*-Means type algorithms in different p-norms is observed in 20 synthetic (GMM) datasets for each combination of feature space and mixed components that are generated. The combination of feature space (V) and mixed components (K), {(v, k)} = {(12, 4), (12, 5), (16, 4), (16, 5), (20, 4)}, where the first element, $v \in V$ and the second, $k \in K$. Each of these dataset contains 1000 data points, equally distributed among the mixed components. These Gaussian mixed model are generated using the MALAB library function called gmm. The gmm has the Euclidean norm, i.e. 2-norm by default.

Therefore, the transformation process, defined in session 4.3.1, is used to translate each of these datasets in different L_p-space. The L_p-space is defined in the range, $\{p_{dist}\}$ = {05, 0.6, ..., 5}. Therefore, in total 20 (number of dataset) times 5 (combination) times 46 (p-norms) equal 5520 number of synthetic datasets are used in the experiments.

The purpose of using datasets defined in different p-norms is to observe the effect of the distance coefficients on four types of *k*-Means algorithms: original *k*-Means, intelligent *k*-Means (*ik*-Means), Minkowski Weighted *k*-Means (*MWk*-Means) and intelligent Minkowski weighted *k*-Means (*iMWk*-Means). The original *k*-Means and *ik*-Means provides no options to alter distance coefficient and inbuilt with distance coefficient equal to 2. But in *MWk*-Means and *iMWk*-Means the distance coefficient, p_{dist}, is parametrized.

Throughout the experiments, *k*-Means and *ik*-Means are executed once with p_plane = 2 whereas *MWk*-Means and *iMWk*-Means are executed 46 times with p_{dist} varying from 0.5 to 5. Therefore, *k*-Means and *ik*-Means are executed 5520 times whereas *MWk*-Means and *iMWk*-Means are executed 46 by 5520 equal 253920 times. In the experiment *k*-Means and *ik*-Means are used as a benchmark. The experimental results from *MWk*-Means and *iMWk*-Means are compared in three ways:

- Using different aggregation functions to compare with *k*-Means and *ik*-Means(see subsection 7.3.1).
- Evaluating how different aggregation functions perform within them (see subsection 7.3.2).
- Assessing how a particular distance coefficient, p_{dist}, behaves with p-norm(p_{plane}) (see subsection 7.3.3).

7.3.1 Performance of Different Aggregation Functions

For each p-norm, *MWk*-Means and *iMWk*-Means are executed 46 time per dataset, whereas *k*-Means and *ik*-Means are executed only once (with distance coefficient 2). Therefore, to compare the performance of *MWk*-Means and *iMWk*-Means with *k*-Means and *iMWk*-means in a particular p-norm, we have to aggregate the results obtained along different distance coefficient, p_{dist}. We have aggregates these results in four ways:

- Average cluster recovery in the range (see figure 7.7).
- Cluster recovery for p_{dist} equals to p_{plane} (see figure 7.8.
- Cluster recovery for p_{dist} equals to 2 (see figure 7.9.

- Maximum cluster recovery in the range (see figure 7.10)*.

* The maximum cluster recovery in the range guides the maximum performance that could be achieved theoretically, as it is only possible when labels are provided. Practically, we can use some cluster recovery index to find the best p_{plane}.

In each sub-section below (from 7.3.1 to 7.3.1), we discuss the four cases listed above using a figure per case. These figures display the performance of the four *k*-Means type algorithms-original *k*-Means, *ik*-Means, *MWk*-Means and *iMWk*-Means- on synthetic datasets generated in different p-norms from 0.5 to 5, with steps of 0.1. X-axes in these figures represent the 'p_{plane}' of the p-norms whereas the y-axes represent the performance of these algorithms (ARI index). Each ARI value for the *k*-Means and *ik*-Means plots is the average of the ARIs obtained from 20 datasets in the particular p-norm defined along the x-axes. Unlike *k*-Means and *ik*-Means where distance coefficient (p_{dist}) is fixed to 2, in *MWk*-Means and *iMWk*-Means the distance coefficient varies from 0.5 to 5 with increments of 0.1. Therefore, for each of the dataset under discussion, *MWk*-Means and *iMWk*-Means are executed 46 times, once per p_{dist} value in the range [0.5, 5]. And the ARI values for the *MWk*-Means and *iMWk*-Means plot at a particular p-norm is the mean of these ARIs obtained from the range of the distance coefficient. As listed above, ARIs values from the p_{dist} range are summarized as: mean, p_{dist} equal to p_{plane} in p-norm, p_{dist} equal to 2 and p_{dist} that maximizes ARIs.

Average Cluster Recovery

In the figure below, ARI values for the *MWk*-Means and *iMWk*-Means for a set of distance coefficients p_{dist} over the range [0.5,5] is summarized as a mean of these values. If we consider these mean ARIs as A1s, then, for a particular 'p' of p-norm along the x-axis, the performance measure along the y-axis is the average of A1s obtained from the 20 datasets.

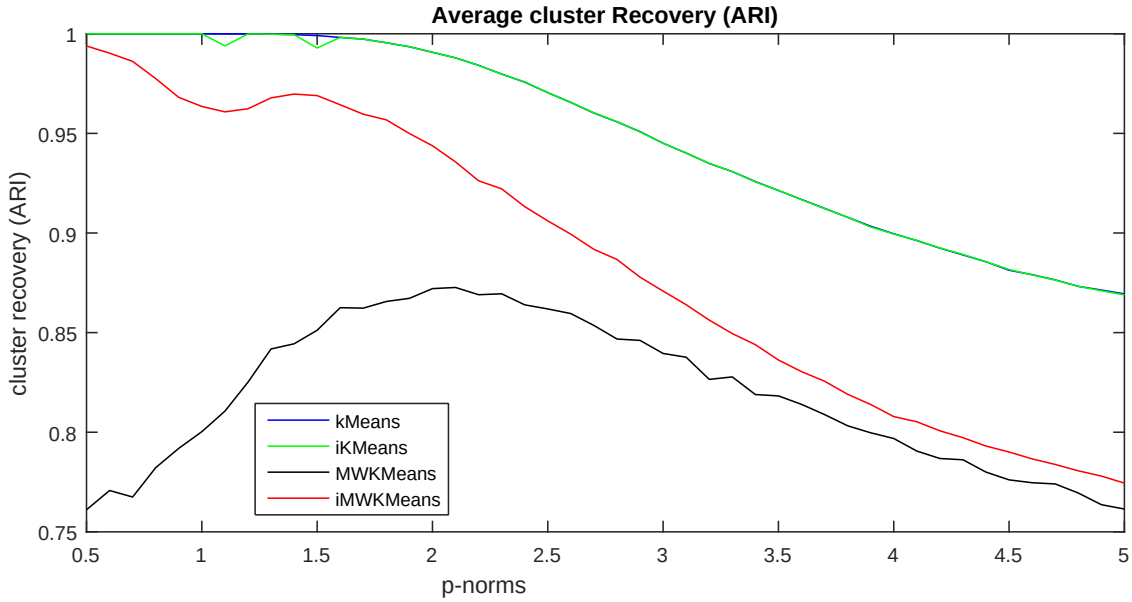


FIGURE 7.7: Performance of four different *k*-Means Type algorithms on GMM datasets defined in different p-norms, p_{plane} ranging from 0.5 to 5.

In figure 7.7, for the p-norms below 1.5, *k*-Means and *ik*-Means provide higher cluster recovery (almost 1) and their performance continually reduces as p-norm space is increased. This look quite obvious from figure 7.6, where it can be seen that when the p-norm is low, GMM clusters are well separated, whereas the clusters get more intermixed when p-norm is increased. In *k*-Means, *ik*-Means and *iMWk*-Means, performance is keep falling down when p-norm is increased, and the trend more smoother in *k*-Means and *ik*-Means. We also observe *ik*-Means shows two sharp decreases at 1.1 and 1.5 (though the decrease is not very significant). As expected, the performance of *MWk*-Means and *iMWk*-Means is not higher at low p-norm space. This is because of the distance coefficient considered within the Minkoswki type of *k*-Means, i.e. each reading in p-norm space is the mean of 46 runs of the particular algorithm where distance coefficient is parametrized in the range of 0.5 to 5. Most significantly, *MWk*-Means perform best when p-norm is around 2.

p_{dist} equals to p_{plane}

One of the options for selecting the distance coefficient, p_{dist} for *MWk*-Means and *iMWk*-Means is the 'p' value of the particular dataset. In the figure below, for each p-norm, ARI values in *MWk*-Means and *iMWk*-Means plots are the average of ARI values obtained from execution of these algorithms on 20 dataset in the particular p-norm when *MWk*-Means and *iMWk*-Means used the distance coefficient equal to the p-norm of the dataset.

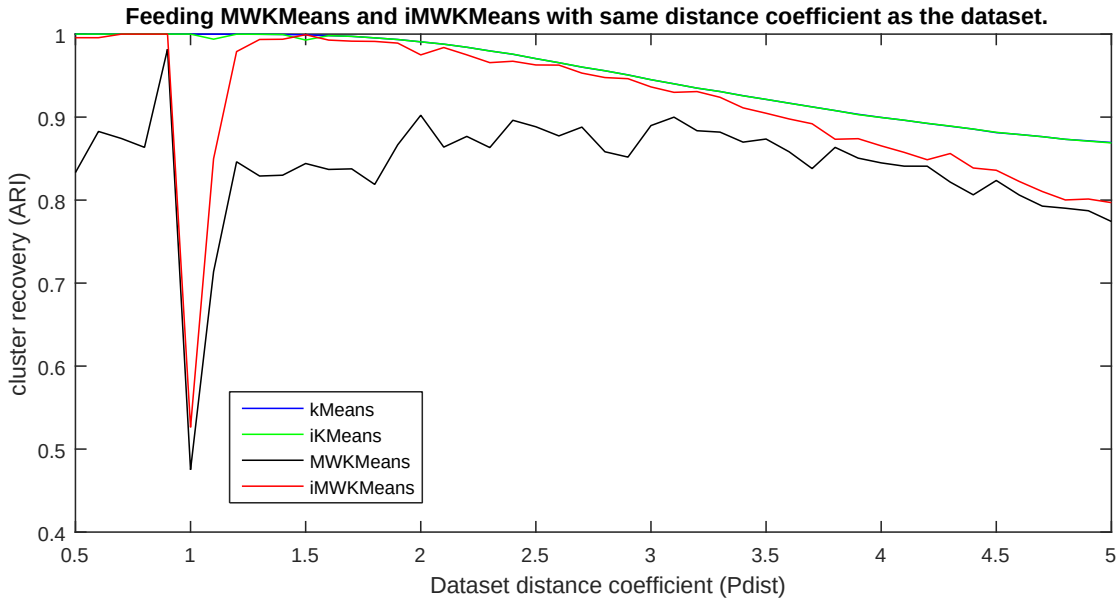


FIGURE 7.8: Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where distance coefficient p_{dist} is equal to p-norm, p_{plane}.

In figure 7.8, for the p-norm below 1.5, *k*-Means and *ik*-Means gives higher cluster recovery (almost 1) and their performance keep decrease as p-norm space is increased. This looks quite obvious, as from figure 7.5, we have seen that when the p-norm is low, GMM clusters are well seperated and the clusters get more intermixed when p-norm is increased. In *k*-Means, *ik*-Means and *iMWk*-Means, performance decreases when p-norm is increased, and the trend is smoother in *k*-Means and *ik*-Means. We also observe that *ik*-Means shows two marked decreases, at 1.1 and 1.5 (though the decrease is not very significant). As expected, the performance of *MWk*-Means and *iMWk*-Means is not higher at low p-norm space. This is because of the distance coefficient considered within the Minkoswki type of *k*-Means i.e. each reading at p-norm space is the mean of 46 runs of the particular algorithm where distance coefficient is parametrized in the range of 0.5 to 5. Most signifacently, *MWk*-Means has best performance when p-norm is around 2.

p_{dist} equals to 2

In the case above, distance coefficient (p_{dist}) for *MWk*-Means and *iMWk*-Means is choosed based on the p-norm, p_{plane} of the dataset. One of the obvious case is to select distance coefficient equal to Euclidean norm i.e. $p_{dist} = 2$. In the figure below, performance, i.e. ARIs values for each p-norm in each plots is the average ARIs obtained in execution of 20 dataset using Euclidean norm.

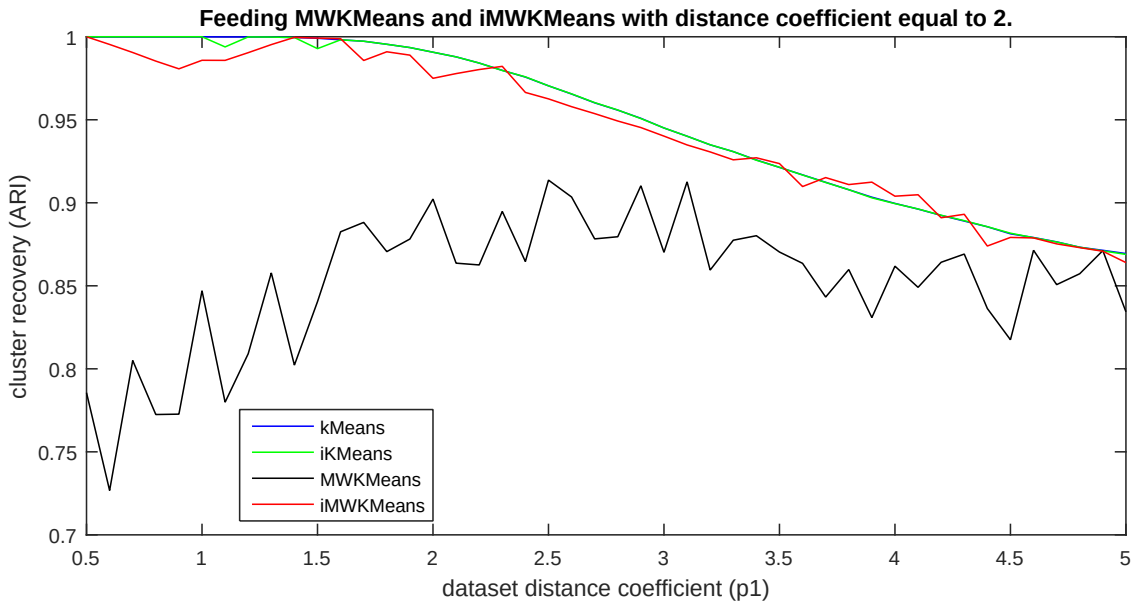


FIGURE 7.9: Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where the distance coefficient p_{dist} is equal to 2.

Performance of *iMWk*-Means is quite similar to that of *k*-Means and *ik*-Means when the distance coefficient,p_{dist} is equal to 2. However, *MWk*-Means shows more deviation in cluster recovery across p-norms and is relatively lower than for other types of *k*-Means.

p_{dist} that yield max ARI

In the figure below, (figure 7.10), for each dataset defined at the particular p-norm, distance coefficient, $p_{dist} \in [0.5, 0.6, ...5]$, is chosen such that it maximizes the ARI. Therefore, each ARI value in the *MWk*-Means and *iMWk*-Means plot below is the average the ARIs obtained from 20 datasets per configuration where *MWk*-Means and *iMWk*-Means are executed with the distance coefficient, p_{dist}, that maximized the objective (ARI).

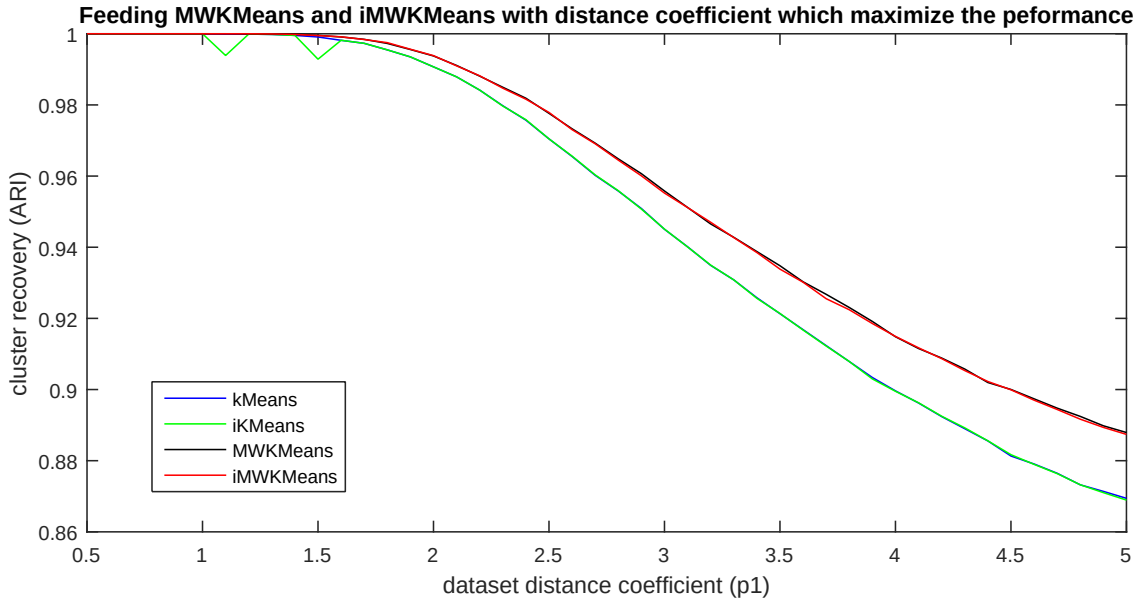


FIGURE 7.10: Performance of different *MWk*-Means and *iMWk*-Means on the GMM defined in different p-norms where the distance coefficient, p_{dist} is considered the one which maximize the ARI.

Plots in the figure above shows that *MWk*-Means and *iMWk*-Means give better performance than *k*-Means and *ik*-Means when *MWk*-Means and *iMWk*-Means use an appropriate distance coefficient, p_{dist}. As expected, performance of all algorithms considered worsens as p-norm is increased. However, the rate in fall of ARIs in significantly lower in the *MWk*-Means and *iMWk*-Means than in *k*-Means and *ik*-Means when an appropriate distance coefficient is provided.

7.3.2 Comparison of Different Aggregation Functions

Four aggregation functions are used to analysis the performance of different distance coefficients, p_{dist}, used in *MWk*-Means and *iMWk*-Means with respect to the datasets generated using a range of p-norms, p_{plane}. Results obtained from the experiments are grouped into two subgroups as represented in two figures below: one from *MWk*-Means (figure 7.11) and the next from *iMWk*-Means (figure 7.12). In these figures, there are four plots, in different colors. Each of these plots represents the performance of *MWk*-Means or *iMWk*-Means under the four aggregation functions: average ARIs over p-norms, the maximum ARIs over different distance coefficient(p_{dist}), average ARIs for distance coefficient (p_{dist}), equal to the dataset p-norm, i.e. $p_{dist} = p_{plane}$ and the average ARIs for distance coefficient, i.e. $p_{dist} = 2$.

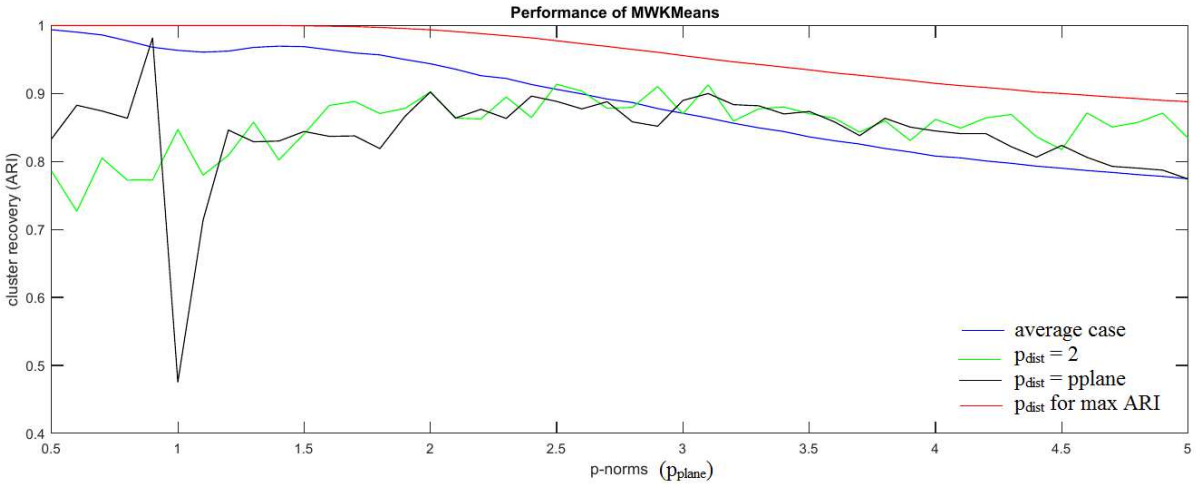


FIGURE 7.11: Performance of four different combinations of the distance coefficient, p_{dist} used in *MWk*-Means (p1) over different datasets defined using a range of p-norms.

Figure 7.11 combines the results (ARIs) from *MWk*-Means for the datasets, using different p-norms in four ways based on the function used to select the distance coefficient for *MWk*-Means. The first plot (in blue) denotes the result when the average ARI is considered for the dataset defined at the particular p-norm. In this case, the distance coefficient for *MWk*-Means is chosen from the range 0.5 to 5 (46 cases) with an interval of 0.1. In the second case, the distance coefficient p_{dist} for *MWk*-Means is equal to 2 (green plot in the figure). The third plot (black in the figure) is the case where the distance coefficient using by *MWk*-Means is equal to p-norm. And the last plot (red) is the one chosen from the range of p_{dist}s such that it yields maximum ARIs for a particular dataset defined at a particular p-norm.

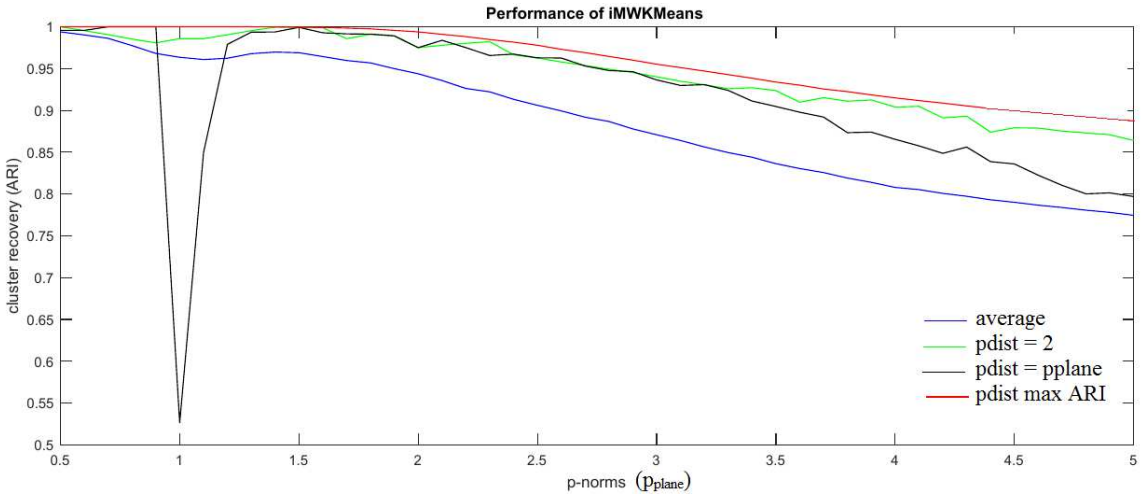


FIGURE 7.12: Performance of four different combination of the distance coefficient, p_{dist} used in *IMWk*-Means for different datasets defined for a range of p-norms, p_{plane}.

Figure 7.12 combines the results (ARIs) from *iMWk*-Means for datasets at different p-norms in four ways based on the function used to select the distance coefficient for *iMWk*-Means.

From figures 7.11 and 7.12, we can conclude that both *MWk*-Means and *iMWk*-Means provide better cluster recovery using distance coefficient, p_{dist} equal to 2 (default value). Also, knowledge of the p-norm of a dataset does not have significant contribution in the performance of *MWk*-Means or *iMWk*-Means as plots for p_{dist} equal p_{plane} is third lowest next to the average case.

7.3.3 p-norms vs Distance Coefficient

Each reading in the table below is the average cluster recovery for each of the algorithms over 100 datasets: 20 datasets for each of the five configurations discussed earlier. The first column in the table represents the particular algorithms used. The second columns contain the distance coefficient used within the particular algorithm. The remaining columns represent the ARI values at particular p-norms, ranging from 0.5 to 5 with interval of 0.5. Only ARI values for certain p-norms are chosen in order to reduce the table width. The first two rows in the table shows the ARI values from *k*-Means and *ik*-Means. Since *k*-Means and *ik*-Means have the Euclidean norm, that is, distance coefficient equal to 2, p_{dist} reading for these row is 2. As discussed earlier, the distance coefficient in *MWk*-Means and *iMWk*-Means can be altered. In the two tables below, for *MWk*-Means and *iMWk*-Means only the ARIs values $p_{dist} \in \{0.5, 1, ..., 5\}$ are displayed. The reason to limiting the readings shown in tables is to synchronise the distance coefficients used in the algorithms with database p-norms.

The table can be read follows: for a particular algorithm in the first column, for a dataset defined in a particular p-norm (columns numbered three to last), each reading down the column shows the ARIs values obtained by the algorithms when the algorithm uses the distance coefficient as listed in the second column. Since *k*-Means and *ik*-Means use the distance coefficient p_{dist} equal to 2, there is only one raw for each of them. The first table shows the results from *MWk*-Means and the second table contains reading from *iMWk*-Means.

The highest ARI values within a column, as given by three of the algorithms (*k*-Means, *ik*-Means and one of *MWk*-Means or *iMWk*-Means) for datasets defined at a particular p-norm, are shown

in bold. This will be of use when comparing the performance of the particular algorithm (*MWk*-Means or *iMWk*-Means) against that of *k*-Means and *ik*-Means. It is also significant to know the best distance coefficient for the particular algorithm (*MWk*-Means or *iMWk*-Means) for a particular p-norm. And, in the two tables below, we mark the reading with an asterisk (*).

p_{dist} vs p_{plane} in *MWk*-Means

As stated above, the table below shows the ARIs for *MWk*-Means (from row three to the last) where rows represent the distance coefficient, p_{dist} used in *MWk*-Means and the columns represent p-norms, the p_{plane} of the dataset.

TABLE 7.1: Performance *MWk*-Means in GMM datasets defined in a range of p-norms.

		GMM distance coefficient (p1)									
		0.5	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**	**3.5**	**4.0**	**4.5**	**5.0**
Algorithms	**p2**	**ARI Index**									
kMeans	2.0	**1.000**	**1.000**	**0.999**	**0.991**	**0.970**	**0.945**	**0.921**	**0.900**	**0.882**	**0.869**
iKMeans	2.0	**1.000**	**1.000**	0.993	0.991	**0.970**	**0.945**	**0.921**	**0.900**	**0.882**	**0.869**
MWKMeans	0.5	*0.833	0.811	0.884	0.908	0.865	0.837	0.779	0.722	0.711	0.682
	1.0	0.760	0.475	0.339	0.210	0.154	0.132	0.108	0.097	0.098	0.093
	1.5	0.736	0.773	0.844	0.875	0.912	0.885	0.869	0.850	*0.861	0.804
	2.0	0.786	0.847	0.840	0.902	0.914	0.870	0.870	0.862	0.818	0.834
	2.5	0.700	0.777	0.848	0.912	0.888	0.869	0.865	0.866	0.825	*0.845
	3.0	0.719	0.766	0.797	0.917	0.900	0.890	*0.888	*0.875	0.859	0.800
	3.5	0.729	0.780	0.895	0.899	0.896	0.885	0.874	0.845	0.841	0.819
	4.0	0.798	0.855	0.883	0.914	*0.919	*0.900	0.861	0.845	0.822	0.796
	4.5	0.795	*0.880	0.888	0.930	0.901	0.870	0.863	0.837	0.824	0.804
	5.0	0.797	0.835	*0.926	*0.933	0.904	0.864	0.842	0.793	0.805	0.774

For all p-norms, *k*-Means and *ik*-Means show better cluster recovery than *MWk*-Means, as we indicated in the figure above, where all entities in the first and second rows are in bold type. For datasets with p-norm equal to 1, larger the distance coefficient (p_{dist}) trigger a reduction in performance. Also, with increasing p-norms, *MWk*-Means shows better performance when the distance coefficient is around 2.

p_{plane} vs p_{dist} in *iMWk*-Means

Again, as in the previous table, the table below shows the ARIs for *iMWk*-Means (from row three to the end) where rows represent the distance coefficient, p_{dist}, used in *iMWk*-Means, and the columns represents p-norms, p_{plane} for the datasets used.

TABLE 7.2: Performance *iMWk*-Means in GMM datasets defined in a range of p-norms.

		GMM distance coefficient (p1)									
		0.5	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**	**3.5**	**4.0**	**4.5**	**5.0**
Algorithms	**p2**	**ARI Index**									
kMeans	2.0	**1.000**	**1.000**	**0.999**	**0.991**	**0.970**	**0.945**	0.921	0.900	0.882	0.869
iKMeans	2.0	**1.000**	**1.000**	0.993	**0.991**	**0.970**	**0.945**	0.921	0.900	0.882	0.869
***iMWk*-Means**	0.5	0.996	***1.000**	0.997	0.969	0.908	0.846	0.789	0.748	0.717	0.688
	1.0	0.781	0.526	0.413	0.313	0.226	0.181	0.140	0.115	0.104	0.095
	1.5	***1.000**	0.986	***0.999**	***0.991**	*0.966	0.939	0.909	0.865	0.854	0.846
	2.0	***1.000**	0.986	***0.999**	0.975	0.963	*0.940	***0.924**	***0.904**	0.879	0.864
	2.5	***1.000**	0.976	***0.999**	0.986	0.963	0.934	0.908	0.903	***0.888**	**0.871**
	3.0	***1.000**	0.972	***0.999**	0.980	0.949	0.936	0.915	0.894	0.883	0.865
	3.5	***1.000**	0.971	0.988	0.971	0.956	0.935	0.905	0.883	0.871	0.855
	4.0	***1.000**	0.961	0.993	0.975	0.945	0.921	0.886	0.865	0.851	0.829
	4.5	***1.000**	0.956	0.987	0.960	0.937	0.915	0.872	0.853	0.836	0.809
	5. 0	***1.000**	0.956	0.969	0.968	0.940	0.883	0.859	0.830	0.808	0.797

Intelligent *MWk*-Means shows the best cluster recovery for datasets defined with p-norm equal to 0.5. Most significantly, *iMWk*-Means gives better cluster recovery than *k*-Means and *ik*-Means for dataset are defined in p-norms greater than 3 and *iMWk*-Means shows better cluster recovery when distance coefficient around to 2 is considered for dataset defined in p-norms greater than one.

Note:
Cluster validation index can be used to find the best distance coefficient (p_{dist}) for both *MWk*-Means and *iMWk*-Means, see results in appendix E.1.

7.4 Conclusion

In this chapter, an algorithm ($pNorm$Transform) is developed which transforms a dataset to different p-norms. This algorithm is practically validated using reverse engineering. With the transformation of a dataset in different p-norms, new features of a dataset can be explored. Moreover, availability of datasets in different p-norms provides data researchers with a large, enriched pole of datasets to conducting simulations.

The range of datasets in different p-norms is used to evaluate the performance of *k*-Means types algorithms. It is observed that when the distance coefficient (p_{dist}) gets smaller, clusters become more widely separated, and hence cluster recovery is increased in *k*-Means type algorithms. It is also observed that the performance of weighted variants of *k*-Means is worst at 1-norm. Feeding the Minkowski variant of weighted *k*-Means with the same distance coefficient as p in p-norm, reduced the performance. However, these variants of *k*-Means have better cluster recovery for dataset in all p-norms which further explore the capability of Minkowski variants of weighted *k*-Means to learn the shape of clusters in different p-norms.

Chapter 8

Extended Experiments

In the previous three chapters, results from the implementation of feature weighting principles in three areas of clustering - p-norm datasets, datasets with missing values and feature selection are discussed in detail. This chapter further extends our knowledge of cluster analysis and aims to answer the following questions:

(i) Are the proposed feature selection algorithms effectively remove noisy features?

(ii) Are the proposed feature selection algorithms suitable for larger datasets?

(iii) Are the proposed feature selection algorithms able to retain the features, required by classifiers?

(iv) Is the performance of proposed feature selection algorithms better in p-norms?

(v) Can partial distance approaches can be implemented with the Minkowski metric in p-norms datasets?

8.1 Experimental set-up

To answer the above questions, experiments are conducted on both the synthetic and real-world (UCI) datasets introduced earlier, in Chapter 3. Research in this chapter is restricted to the added noisy features, where noisy features are generated from the Uniform distribution. Three cases of added noisy features - no noise, half noise and full noise - are analysed. The addition of noisy features in the tables below is denoted by the datasetName+*K*NF, where K represents either 0 or $\frac{\|V\|}{2}$ or $\|V\|$ number of added noisy features with $\|V\|$ feature space cardinality. For example, 1000x8-2+4NF represents a dataset with two Gaussian clusters, 1000 data points and 8 original features. The dataset (1000x8-2) has 4 more noisy features added on it.

In Chapter 5, the intelligent *k*-Means for feature selection, iKFS, is observed to have no role in the selection of features, hence, iKFS is discarded. Therefore, in addition to our proposed feature selection methods, feature similarity based on feature selection, FSFS, and multi-cluster based feature selection, MCFS, are analysed further.

As in the earlier experiments, the Silhouette index is used for parameter tuning in FSFS and MCFS (see [2] for further detail). In FSFS and MCFS, the best case is chosen by using the label information available (supervised learning). The first three questions set-up above - (i), (ii) and (iii) - are analysed by setting the distance coefficient (p) to 2.

In earlier chapters, Chapter 5 and Chapter 6, the value of p lies between 1.1 to 5.0, with an interval of 0.1. However, to observe the trend of p, a larger interval of 0.5 can be considered. Therefore, to answer questions (iv) and (v), the values of p_{plane} for p-norm and the coefficient of the Minkowski metric, p_dist, in *mean*-FSFW and *max*-FSFW are set to {1.1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5,

5}.

Experiments are run in both the synthetic and real-world (UCI) datasets to answer questions(i) and (iii). Two datasets from the UCI repository which have higher feature space cardinality, i.e., $\|V\| \geq 100$, are considered, for question (ii). To answer questions (iv) and (v), experiments are conducted in the synthetic datasets.

8.2 Extension of feature selection

In Chapter 5, performance of the proposed feature selection algorithms is observed on the basis of cluster recovery and percentage of the feature selection. In [2], we published the results of using the feature selection algorithm on the basis of the percentage of noisy feature selected and have extended the work in two larger datasets from the UCI repository.

8.2.1 Percentage of noisy feature selection

In the two tables 8.1 and 8.2, results from synthetic and the real-world (UCI) datasets are summarized, showing the percentage of noisy features selected.

Synthetic datasets

TABLE 8.1: The average percentage of noisy features selected, and their respective standard deviations, in synthetic datasets

		Accuracy					
		FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
	Dataset	SIL	Best Case	SIL	Best Case	*mean*-FSFW	*max*-FSFW
avg % noisy feat. sel.	1000x8-2 +4NF	73.500/31.78	51.000/44.71	93.000/24.00	79.000/39.80	**0.000/0.00**	7.500/16.77
	1000x8-2 +8NF	73.750/35.64	44.500/42.22	89.750/24.96	76.500/36.02	**0.500/2.45**	4.250/9.56
	1000x12-3 +6NF	80.000/33.50	63.333/43.33	95.666/9.89	91.333/21.15	**0.000/0.00**	1.000/5.17
	1000x12-3 +12NF	81.001/32.62	60.667/43.69	85.834/18.50	77.667/23.95	**0.000/0.00**	0.833/3.00
	1000x16-4 +8NF	89.750/22.32	81.750/34.02	82.250/14.60	80.250/14.16	**0.000/0.00**	**0.000/0.00**
	1000x16-4 +16NF	87.875/27.28	73.125/40.80	69.500/20.60	62.250/20.54	**0.000/0.00**	0.500/2.11
	1000x20-5 +10NF	83.400/32.16	77.600/37.18	47.000/22.11	42.000/18.11	**0.000/0.00**	**0.000/0.00**
	1000x20-5 +20NF:20	78.700/36.83	66.400/42.24	43.600/29.33	32.300/20.40	**0.000/0.00**	**0.000/0.00**

mean-FSFW is able to remove almost all added noisy features from the Gaussian clusters. *max*-FSFW is able to remove most of the noisy features for larger datasets and up to 92.5% of noisy features in the Gaussian clusters with feature space cardinality of 8+4NF. FSFS and MCFS are hardly able to remove any noisy features as the maximum percentage of noisy features removed is around 70% .

Real-world (UCI) datasets

mean-FSFW is able to remove all noisy features from eight out of ten UCI datasets. In the Car Evaluation and the Tic-Tac-Toe datasets, *mean*-FSFW is not able to remove a single noisy features. In comparison, *max*-FSFW is more effective than *mean*-FSFW, as *max*-FSFW is able to remove almost all noisy features from ten UCI datasets except the Car Evaluation dataset with full noise.

Under supervised learning, i.e. “Best Case", MCFS is also effective for removing noisy features. However, performance is significantly reduced when the Silhouette index is used for parameter tuning, i.e. unsupervised learning. FSFS is least effective in removing the noisy features in both

TABLE 8.2: The average percentage of noisy features selected, in the real-world (UCI) datasets.

		Accuracy					
		FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
Data Type	Dataset	SIL	Best Case	SIL	Best Case	*mean*-FSFW	*max*-FSFW
Integer/Real	Ecoli +4NF	100.000	25.000	100.000	25.000	**0.000**	**0.000**
	Ecoli +8NF	87.500	0.000	100.000	25.000	**0.000**	**0.000**
	Glass +5NF	80.000	0.000	100.000	0.000	**0.000**	**0.000**
	Glass +10NF	80.000	0.000	100.000	0.000	**0.000**	**0.000**
	Ionosphere +17NF	5.880	0.000	**0.000**	0.000	**0.000**	**0.000**
	Ionosphere +34NF	2.940	0.000	**0.000**	0.000	**0.000**	**0.000**
	Iris +2NF	100.000	0.000	100.000	0.000	**0.000**	**0.000**
	Iris +4NF	100.000	0.000	100.000	0.000	**0.000**	**0.000**
	Wine +7NF	100.000	0.000	85.710	85.710	**0.000**	**0.000**
	Wine +14NF	100.000	28.570	85.710	50.000	**0.000**	**0.000**
Mix	Austra C.A.+21NF	100.000	100.000	**0.000**	0.000	**0.000**	**0.000**
	Austra C.A.+42NF	100.000	100.000	40.480	0.000	**0.000**	**0.000**
	Teaching Assistant +28NF	**0.000**	14.290	**0.000**	0.000	**0.000**	**0.000**
	Teaching Assistant +56NF	**0.000**	5.360	**0.000**	0.000	**0.000**	**0.000**
	Zoo +8NF	100.000	0.000	62.500	0.000	**0.000**	**0.000**
	Zoo +16NF	100.000	0.000	75.000	0.000	**0.000**	**0.000**
Categorical	Car Evaluation +11NF	9.090	9.090	**0.000**	9.090	100.000	**0.000**
	Car Evaluation +22NF	22.730	4.550	**0.000**	0.000	100.000	100.000
	Tic Tac Toe: +14NF	28.570	7.140	92.860	0.000	100.000	**0.000**
	Tic Tac Toe +28NF	53.570	3.570	**0.000**	0.000	100.000	**0.000**

supervised and unsupervised learning. FSFS with its Best Case (supervised learning) is far better at removing noisy features than the unsupervised learning method.

mean-FSFW fails to remove redundant features from Car Evaluation and Tic-Tac-Toe dataset. However, *max*-FSFW is able to all reduce redundant features except in one case- 'Car Evaluation + 22NF'. The Car Evaluation has six and Tic-Tac-Toe dataset has nine features which are all categorical, this shows that *max*-FSFW is better than *mean*-FSFW when a dataset has all categorical features.

8.2.2 Extending the method to larger datasets

To observe the performance of our proposed feature selection algorithms for larger datasets, the "Low Resolution Spectrometer" and "Glass Sensor Array Batch10" datasets are considered. The "Low Resolution Spectrometer" dataset has 101 features, whereas the "Gas Sensor Array Batch10" dataset has 129 features. In the "Low Resolution Spectrometer" dataset, 51 and 65 noisy features are added to create half and full noise datasets, whereas, in "Gas-Sensor-Array-Batch10" dataset, 65 and 129 noisy features are added. All noisy features are drawn from the Uniform distribution. The performance of the algorithms are observed on the basis of cluster recovery (ARI) and the percentage of noisy feature selected.

TABLE 8.3: Performace of feature selection algorithms in UCI datasets with a higher degree of feature space cardinality.

		Accuracy						
			FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
	Dataset	*k*-Means	SIL	Best Case	SIL	Best Case	*mean*-FSFW	*max*-FSFW
cluster recovery (ARI)	Low Resolution Spectrometer	0.202	0.210	0.230	0.240	0.260	0.260	**0.270**
	Low Resolution Spectrometer +51NF	0.207	0.180	0.220	0.180	0.240	0.220	**0.280**
	Low Resolution Spectrometer +101NF	0.194	0.180	0.200	0.190	0.230	**0.230**	0.070
	Gas Sensor Array Batch10	0.126	0.120	0.210	0.170	0.170	**0.190**	0.140
	Gas Sensor Array Batch10 +65NF	0.128	0.130	0.210	0.130	0.190	0.120	**0.140**
	Gas Sensor Array Batch10 +129NF	0.126	**0.140**	0.210	**0.140**	0.160	0.130	**0.140**
% of noisy feature selected	Low Resolution Spectrometer +51NF	–	100.000	100.000	98.040	25.490	**0.000**	**0.000**
	Low Resolution Spectrometer +101NF	–	100.000	100.000	97.030	48.510	**0.000**	100.000*
	Gas Sensor Array Drift Batch10 +65	–	78.290	100.000	**0.000**	**0.000**	**0.000**	**0.000**
	Gas Sensor Array Drift Batch10 +129	–	83.720	100.000	**0.000**	**0.000**	**0.000**	**0.000**

- Cluster recovery (ARI)

  mean-FSFW and *max*-FSFW are able to improve the cluster recovery for both datasets in most cases, with and without added noise, since the ARI index is better in all cases compared to *k*-Means (without feature selection). *mean*-FSFW is the best for the “Low Resolution Spectrometer" dataset with higher noise and the “Gas Sensor Array Batch10" dataset when there are no noisy features. In most cases, *mean*-FSFW and *max*-FSFW are able to match the cluster recovery index (ARI). In some cases, *mean*-FSFW and *max*-FSFW are even better than the “Best Case" of FSFS and MCFS.

- Percentage of noisy feature selected

  mean-FSFW is able to remove all added noisy features in both datasets. *max*-FSFW also gives the same results, except with the “Low Resolution Spectrometer" dataset with full noise. MCFS is also able to remove all added noisy features from the “Gas Sensor Array Drift Batch10" dataset in supervised and unsupervised learning. When compared with FSFS, it is better at removing added noisy features in the “Low Resolution Spectrometer" dataset, as FSFS has no contribution to make in removing the added noisy features.

Note*
With the addition of full noise, *max*-FSFW is not able to remove any noisy features in Low-Resolution Spectrometer, though it can remove all noisy features when there is half noise. Original features are likely more random such that the addition of full noise has minimized the feature weighting index between them.

8.2.3 Performance of classifiers after feature selection

Feature selection algorithms are normally considered as a first step in data mining and are used before classification. Feature selection algorithms are expected to retain the “important" features required for classification. Therefore, the effect of proposed feature selection algorithms in three classifiers - KNN, Decision tree and Navie bayes - are observed with 10-fold cross-validation on both synthetic and the real-world (UCI) datasets. The performance of these classifiers is measured using the ARI score before and after the feature selection. In KNN, the value of K is set to the square root of the number of entities in the dataset.

Table 8.4 and Table 8.5 show the average ARI score by the classifiers in synthetic and the real-world datasets with and without feature selection.

Synthetic datasets

TABLE 8.4: The average accuracy of three classifiers, and respective standard deviations, on synthetic data sets containing about 50% or 100% extra features composed of uniformly random noise.

	KNN			Decision Tree			navie Bayes		
Datasets	Orginal	After meanFWFS	After maxFWFS	Original	After meanFWFS	After maxFWFS	Original	After meanFWFS	After maxFWFS
1000x12-3-NF:12	0.954/0.07	**0.985**/0.03	0.980/0.05	0.889/0.07	**0.891**/0.07	**0.891**/0.08	**0.986**/0.03	**0.986**/0.03	0.982/0.05
1000x12-3-NF:6	0.973/0.05	**0.984**/0.03	0.974/0.05	0.890/0.07	**0.892**/0.07	**0.892**/0.08	**0.986**/0.03	**0.986**/0.03	0.976/0.05
1000x12-3-NF:0	**0.984**/0.03	0.948/0.09	0.917/0.13	**0.891**/0.07	0.876/0.11	0.853/0.14	**0.986**/0.03	0.950/0.09	0.920/0.13
1000x16-4-NF:16	0.972/0.02	**0.996**/0.01	0.994/0.01	0.885/0.05	**0.886**/0.05	**0.886**/0.05	**0.996**/0.01	**0.996**/0.01	0.995/0.01
1000x16-4-NF:8	0.987/0.01	**0.995**/0.01	0.990/0.01	0.886/0.05	0.887/0.05	**0.888**/0.05	**0.996**/0.01	**0.996**/0.01	0.991/0.01
1000x16-4-NF:0	**0.996**/0.01	0.977/0.03	0.948/0.06	**0.887**/0.05	0.885/0.05	0.856/0.07	**0.996**/0.01	0.978/0.02	0.950/0.06
1000x20-5-NF:20	0.979/0.02	**0.999**/0.00	0.998/0.00	0.887/0.04	0.887/0.04	**0.888**/0.04	**0.999**/0.00	**0.999**/0.00	**0.999**/0.00
1000x20-5-NF:10	0.992/0.01	**0.999**/0.00	0.996/0.01	**0.888**/0.04	**0.888**/0.04	0.887/0.04	**0.999**/0.00	**0.999**/0.00	0.996/0.01
1000x20-5-NF:0	**0.999**/0.00	0.989/0.01	0.976/0.02	**0.888**/0.04	0.882/0.04	0.870/0.04	**0.999**/0.00	0.991/0.01	0.978/0.02
1000x8-2-NF:8	0.943/0.07	**0.967**/0.05	0.960/0.06	0.892/0.1	0.896/0.1	**0.899**/0.1	0.970/0.04	**0.971**/0.04	0.966/0.05
1000x8-2-NF:4	0.960/0.05	**0.968**/0.05	0.945/0.09	0.895/0.1	**0.897**/0.1	0.890/0.12	**0.971**/0.04	0.970/0.04	0.949/0.09
1000x8-2-NF:0	**0.967**/0.05	0.934/0.09	0.899/0.13	**0.897**/0.1	0.884/0.12	0.852/0.16	**0.971**/0.04	0.936/0.09	0.901/0.13

Real-world (UCI) datasets

TABLE 8.5: The average accuracy of three classifiers, and their respective standard deviations, in real-world data sets containing about 50% or 100% extra features composed of uniformly random noise.

		KNN			Decision Tree			navie Bayes		
	Datasets Configuration	Orginal	After meanFWFS	After maxFWFS	Original	After meanFWFS	After maxFWFS	Original	After meanFWFS	After maxFWFS
Integer/Real	Ecoli-336x7-8-NF:8	**0.513**/0.02	0.373/0.01	0.157/0.01	**0.675**/0.03	0.274/0.02	0.151/0.01	**0.515**/0.02	0.368/0.01	0.160/0.01
	Ecoli-336x7-8-NF:4	**0.597**/0.01	0.159/0.01	0.000/0.00	**0.680**/0.02	0.153/0.01	0.038/0.00	**0.596**/0.01	0.156/0.01	0.000/0.00
	Ecoli-336x7-8-NF:0	**0.757**/0.01	0.000/0.00	0.000/0.00	**0.712**/0.02	0.038/0.00	0.038/0.00	**0.757**/0.01	0.000/0.00	0.000/0.00
	Glass-214x9-6-NF:10	0.146/0.01	0.273/0.02	**0.327**/0.02	0.288/0.03	0.332/0.03	**0.373**/0.04	0.144/0.01	0.274/0.02	**0.325**/0.02
	Glass-214x9-6-NF:5	0.161/0.01	0.277/0.02	**0.321**/0.02	0.332/0.03	0.336/0.03	**0.354**/0.03	0.161/0.01	0.276/0.01	**0.313**/0.02
	Glass-214x9-6-NF:0	0.284/0.02	0.234/0.02	**0.288**/0.02	**0.340**/0.03	0.332/0.03	0.321/0.03	0.281/0.02	0.234/0.02	**0.287**/0.02
	Ionosphere-351x33-2-NF:34	0.206/0.01	**0.551**/0.02	0.209/0.00	0.493/0.03	**0.543**/0.04	0.209/0.00	0.206/0.01	**0.549**/0.02	0.209/0.00
	Ionosphere-351x33-2-NF:17	0.306/0.02	**0.366**/0.02	0.209/0.00	0.540/0.03	**0.586**/0.03	0.209/0.00	0.299/0.02	**0.364**/0.01	0.209/0.00
	Ionosphere-351x33-2-NF:0	0.437/0.02	**0.520**/0.02	0.289/0.02	0.572/0.03	**0.573**/0.02	0.357/0.04	0.432/0.02	**0.518**/0.02	0.288/0.02
	Iris-150x4-3-NF:4	0.713/0.02	0.876/0.02	**0.882**/0.02	0.839/0.03	**0.852**/0.03	0.849/0.02	0.711/0.02	**0.883**/0.02	0.881/0.02
	Iris-150x4-3-NF:2	0.741/0.02	0.885/0.02	**0.886**/0.00	0.852/0.02	0.844/0.02	**0.869**/0.02	0.747/0.02	0.881/0.02	**0.885**/0.00
	Iris-150x4-3-NF:0	0.880/0.02	0.884/0.01	**0.885**/0.00	0.847/0.02	**0.869**/0.02	0.860/0.03	**0.886**/0.02	0.885/0.00	0.884/0.00
	Wine-178x13-3-NF:14	0.730/0.03	**0.913**/0.01	0.891/0.02	0.714/0.04	**0.719**/0.04	0.710/0.04	0.741/0.03	**0.917**/0.01	0.892/0.02
	Wine-178x13-3-NF:7	0.846/0.02	0.836/0.02	**0.869**/0.02	0.714/0.03	0.724/0.04	**0.753**/0.04	0.845/0.03	0.838/0.02	**0.867**/0.01
	Wine-178x13-3-NF:0	**0.914**/0.02	0.876/0.02	0.869/0.01	0.709/0.04	**0.770**/0.04	0.768/0.04	**0.914**/0.01	0.877/0.02	0.872/0.01
Mix	AustraCC-690x42-2-NF:42	**0.512**/0.01	0.233/0.01	0.491/0.02	0.360/0.03	0.179/0.02	**0.399**/0.03	**0.507**/0.01	0.236/0.01	0.495/0.01
	AustraCC-690x42-2-NF:21	0.525/0.01	0.547/0.01	**0.554**/0.00	0.367/0.02	0.410/0.02	**0.432**/0.03	0.527/0.01	0.547/0.01	**0.555**/0.01
	AustraCC-690x42-2-NF:0	**0.507**/0.01	0.490/0.01	0.506/0.01	0.413/0.03	0.412/0.02	**0.439**/0.02	**0.509**/0.01	0.491/0.01	0.508/0.01
	TeachingAssistant-151x56-3-NF:56	**0.071**/0.02	0.029/0.02	0.034/0.01	-0.002/0.01	**0.061**/0.01	0.056/0.02	**0.070**/0.02	0.035/0.01	0.033/0.01
	TeachingAssistant-151x56-3-NF:28	**0.079**/0.02	0.038/0.01	0.040/0.01	-0.003/0.01	0.082/0.02	**0.083**/0.02	**0.076**/0.02	0.040/0.01	0.042/0.01
	TeachingAssistant-151x56-3-NF:0	**0.049**/0.01	0.000/0.00	-0.002/0.00	**0.105**/0.03	0.045/0.01	0.020/0.01	**0.051**/0.01	0.000/0.00	0.000/0.01
	Zoo-101x16-7-NF:16	0.946/0.00	**0.958**/0.00	0.893/0.01	0.938/0.01	**0.958**/0.00	0.955/0.00	0.944/0.00	**0.958**/0.01	0.893/0.01
	Zoo-101x16-7-NF:8	0.942/0.01	**0.958**/0.00	**0.958**/0.00	0.941/0.00	**0.958**/0.00	**0.958**/0.00	0.942/0.01	**0.958**/0.00	**0.958**/0.00
	Zoo-101x16-7-NF:0	0.932/0.00	**0.954**/0.01	0.933/0.00	0.948/0.00	**0.958**/0.00	0.957/0.01	0.932/0.00	**0.955**/0.01	0.933/0.00
Categorical	CarEvaluation-1728x21-4-NF:22	**0.552**/0.01	0.037/0.01	0.037/0.01	**0.806**/0.02	0.122/0.02	0.124/0.01	**0.554**/0.01	0.038/0.01	0.039/0.01
	CarEvaluation-1728x21-4-NF:11	**0.594**/0.01	0.000/0.00	0.000/0.00	**0.813**/0.01	0.018/0.01	0.000/0.00	**0.595**/0.01	0.000/0.00	0.000/0.00
	CarEvaluation-1728x21-4-NF:0	**0.439**/0.01	0.396/0.01	0.397/0.01	**0.878**/0.01	0.400/0.01	0.402/0.00	**0.439**/0.00	0.397/0.01	0.396/0.00
	TicTacToe-958x27-2-NF:28	**0.487**/0.01	0.004/0.01	0.020/0.01	**0.427**/0.04	0.000/0.01	0.023/0.01	**0.483**/0.02	0.004/0.01	0.021/0.01
	TicTacToe-958x27-2-NF:14	**0.527**/0.02	0.014/0.01	0.000/0.00	**0.574**/0.03	0.005/0.01	0.043/0.01	**0.528**/0.02	0.013/0.01	0.000/0.00
	TicTacToe-958x27-2-NF:0	**0.107**/0.01	0.004/0.00	0.002/0.00	**0.731**/0.03	0.014/0.01	0.014/0.01	**0.104**/0.01	0.003/0.01	0.002/0.00

In the Table 8.5, results are grouped based on the attribute types- integer/real, mix and categorical. When a dataset has integral/real or mix attribute types, *mean*-FWFS and *max*-FSFW are able to retain the true or important features required by classification algorithm, in most of the dataset except Ecoli. Whereas, when dataset have all categorical attributes (here Car Evaluation and TicTacToe dataset) both *mean*-FWFS and *max*-FWFS are not able to retain the important features as ARIs are significantly dropped in these cases.

In case feature selection is necessary before classification, for real-world dataset, meanFWFS is a better option than maxFWFS for decision tree and maxFWFW is better than meanFWFS for navie bayes and KNN classifiers.

8.2.4 Performance of feature selection in L_p space

One of the areas of our research is analysis the performance of the cluster recovery algorithms in p-norms. In the table below, the effects of four feature selection algorithms on cluster recovery are observed in Gaussian clusters defined in different p-norms. The synthetic datasets considered earlier are translated into different p-norms, where $p_{plane} \in \{1.1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

The Silhouette index is used to identify best parameters in FSFS and MCFS. In the previously described cases, the distance coefficient (p_{dist}) is set to 2 in *mean*-FSFW and *max*-FSFW. For this experiment, the value of $p_{dist} \in \{1.1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$. The Silhouette index is used to find the best value of the distance coefficient, p_{dist}, in *mean*-FSFW and *max*-FSFW for a dataset defined in p_{plane}-norm.

TABLE 8.6: The average adjusted Rand Index and its standard deviation for the cluster recovery in synthetic datasets with and without noisy features in different p-norms. Silhouette Index is used for parameter tunning.

	Accuracy											
	No Noise				Half Noise				Full Noise			
p_1-norm	FSFS	MCFS	*mean*-FSFW	*max*-FSFW	FSFS	MCFS	*mean*-FSFW	*max*-FSFW	FSFS	MCFS	*mean*-FSFW	*max*-FSFW
1.1	0.169/0.29	0.271/0.27	0.697/0.33	**0.793**/0.27	0.116/0.22	0.425/0.27	**0.921**/0.17	0.852/0.22	0.116/0.22	0.398/0.28	**0.976**/0.05	0.958/0.07
1.5	0.143/0.26	0.288/0.27	0.716/0.33	**0.842**/0.24	0.106/0.2	0.444/0.27	**0.918**/0.16	0.883/0.19	0.106/0.2	0.387/0.27	**0.976**/0.05	0.939/0.12
2	0.144/0.26	0.276/0.27	0.709/0.32	**0.879**/0.2	0.085/0.15	0.44/0.28	**0.913**/0.17	0.852/0.2	0.085/0.15	0.417/0.28	**0.975**/0.07	0.943/0.11
2.5	0.167/0.29	0.291/0.27	0.683/0.33	**0.851**/0.24	0.082/0.16	0.45/0.28	**0.934**/0.12	0.878/0.15	0.082/0.16	0.429/0.27	**0.979**/0.04	0.952/0.12
3	0.167/0.29	0.316/0.29	0.728/0.31	**0.865**/0.22	0.098/0.18	0.435/0.27	**0.888**/0.21	0.871/0.18	0.109/0.2	0.416/0.27	**0.974**/0.06	0.948/0.12
3.5	0.188/0.32	0.293/0.28	0.753/0.3	**0.845**/0.23	0.09/0.15	0.436/0.27	**0.913**/0.16	0.845/0.18	0.098/0.18	0.409/0.27	**0.974**/0.06	0.953/0.08
4	0.19/0.31	0.312/0.28	0.757/0.29	**0.842**/0.23	0.094/0.17	0.444/0.27	**0.899**/0.19	0.892/0.17	0.094/0.17	0.415/0.27	**0.977**/0.05	0.949/0.09
4.5	0.191/0.31	0.29/0.28	0.794/0.27	**0.875**/0.19	0.094/0.17	0.427/0.27	**0.886**/0.2	0.874/0.17	0.094/0.17	0.42/0.27	**0.976**/0.06	0.966/0.07
5	0.188/0.31	0.307/0.28	0.74/0.31	**0.852**/0.22	0.099/0.17	0.428/0.26	**0.88**/0.2	0.875/0.18	0.099/0.17	0.395/0.26	**0.977**/0.06	0.956/0.09

The table above clearly shows that both *mean*-FSFW and *max*-FSFW outperform FSFS and MCFS in the Gaussian mixed-model, defined in different p-norms using unsupervised learning with or without added noisy features. Moreover, *max*-FSFW is better than *mean*-FSFW for all p-norms when there are no noisy features added in the Gaussian clusters. With the addition of the noisy features, *mean*-FSFW is slightly better than *max*-FSFW.

8.3 Extension of partial distance to the Minkowski metric

The partial distance approach used in Chapter 6, as an alternative to imputation methods, shows promising results. The partial distance approach is further extended to the Minkowski metric in a way that the proposed feature selection algorithms can address the missing values implicitly. The extended algorithms are called *mean*FSFWext*PD* and *max*FSFWext*PD*.

For the experimental set-up, the missing values are introduced in the first two features of the synthetic datasets used in earlier experiments. The missing values are missing completely at random, MCAR, and the percentage of the missing values is set to 40%. The performance of the extended algorithms is compared with three imputation methods used in earlier chapter (Chapter 6) and are further experimented in Gaussian clusters, defined in different p-norms. The performance is measured based on the adjusted Rand Index.

Note:
The code for the proposed *mean*FSFWext*PD* and *max*FSFWext*PD* are available in the repository at https://bitbucket.org/m-learning/phd/src/master/.

TABLE 8.7: The average adjusted Rand Index and its standard deviation for the cluster recovery in synthetic datasets defined in p-norms. The datasets have missing values in first two features generated using the missing completely at random (MCAR) mechanism. The Performance columns contains mean/std of the ARIs.

	Performance of Algorithm							
	Imputation					Partial Distance		
	average		KNN		C Regression		Partial Distance	
p	*mean*FSFW	*max*FSFW	*mean*FSFWext	*max*FSFWext	*mean*FSFWext	*max*FSFWext	*mean*FSFWext*PD*	*max*FSFWext*PD*
1.1	0.54/0.34	**0.812**/0.27	0.435/0.34	0.699/0.31	0.453/0.33	0.725/0.29	0.468/0.38	0.639/0.37
1.5	0.892/0.15	**0.931**/0.13	0.839/0.19	0.89/0.16	0.841/0.19	0.885/0.16	0.88/0.16	0.886/0.17
2	0.892/0.15	0.899/0.15	0.859/0.18	0.875/0.18	0.856/0.19	0.861/0.19	0.914/0.13	**0.916**/0.14
2.5	0.906/0.14	**0.923**/0.14	0.866/0.19	0.881/0.18	0.862/0.19	0.892/0.17	0.902/0.15	0.923/0.14
3	0.9/0.14	**0.939**/0.12	0.862/0.19	0.893/0.16	0.853/0.19	0.9/0.18	0.911/0.14	0.906/0.15
3.5	0.884/0.16	0.926/0.13	0.849/0.2	0.884/0.17	0.836/0.2	0.885/0.17	0.902/0.14	**0.933**/0.13
4	0.911/0.14	**0.917**/0.13	0.889/0.16	0.893/0.17	0.88/0.18	0.894/0.17	0.912/0.14	0.888/0.17
4.5	0.915/0.14	0.919/0.14	0.9/0.17	0.902/0.16	0.877/0.18	0.874/0.19	**0.925**/0.12	0.923/0.14
5	0.912/0.14	**0.92**/0.13	0.876/0.18	0.908/0.15	0.868/0.19	0.875/0.19	0.892/0.17	0.904/0.15

- p-norm, $p < 2$

  Average imputation is found to be the best options for addressing missing values in *mean*-FSFW and *max*-FSFW for the Gaussian clusters, defined with p-norms, where $p < 2$.

- p-norm, $p = 2$

  When the Gaussian clusters are defined in 2-norm, the partial distance approach is found to be the best in both proposed feature selection algorithms.

- p-norm, $p > 2$

  For the Gaussian clusters, defined in p-norms, where $p > 2$, an average imputation method is the best option for replacing missing values, closely followed by the partial distance approaches for the proposed feature selection algorithms.

8.4 Conclusion

In this chapter, extended experiments are conducted in the three different areas of cluster analysis carried out in previous chapters, and are combined together. It has been shown that the proposed feature selection algorithms are able to remove noisy features from both synthetic and

the real-world datasets more effectively. The proposed feature selection algorithms are also suitable for the selection of the “informative" features from larger datasets and are able to retain the “true" features, required by classifiers.

Moreover, the proposed feature selection algorithms provide the best alternative for feature selection of Gaussian clusters, defined in different p-norms. The partial distance approach is found to be the best alternative to imputation methods for addressing missing values in Gaussian clusters, defined in 2-norm.

Chapter 9

Discussion and Conclusion

9.1 Research outcomes

The main objective of this thesis was to explore feature weighting techniques in cluster analysis through Minkowski metric and use them for feature selection. This thesis took advantage of prior work in the Minkowski weighted variant of *k*-Means and observed how missing values can be handled in the weighted variants of *k*-Means. Also, the effectiveness of cluster analysis in L_p space through *k*-Means type algorithms was explored.

There are three major approaches for handling missing values: removing instances with missing values, replacing the missing values with some known values, and implicitly modifying algorithms so that missing values are handled within themselves. The last two approaches are explored in this thesis. Based on the previous work on addressing missing values in *k*-Means algorithm types [112, 163], these ideas are extended to the weighted and Minkowski variants of *k*-Means and their effectiveness explored with values missing completely at random (MCA) and not missing at random (NMAR) mechanisms, where the missing values were presented in a feature with lowest and highest correlation with respect to each feature's dataset. The other approach, no imputation, requires a change in the distance metric within the algorithms so that the missing values are handled implicitly.

In the literature [104, 164, 165], distance metrics used within the algorithms are modified such that no imputation is required to address missing values. Here, the distance metric was successfully replaced by a partial-distance metric within both the weighted and Minkowski *k*-Means frameworks to address the missing value problem in the proposed feature selection algorithms. Good results are found on both synthetic and UCI datasets.

Earlier work introduced the Minkowski metric in Minkowski weighted *k*-Means[92, 162, 137, 2] variants to find regular shaped clusters other than spherical. However, the usefulness of the Minkowksi metric had yet to be fully explored since the experimental datasets use the 2-norm. A translation procedure was created which is successfully used to translate a dataset from an original p_{old}-norm, usually 2-norm, to a new p_{new}-norm. The p-norm datasets are then used to explore the behaviour of *k*-Means type algorithms including the Minkowski variants. Experimental results in this thesis demonstrate that Minkowski weighted *k*-Means is equally effective in datasets defined in different p-norms.

The main results of this thesis are:

i) A partial distance variant of weighted *k*-Means extended to partial distance (*Wk*-Means*PD*), intelligent weighted *k*-Means (*iWk*-Means*PD*) extended to partial distance, Minkowski Weighted *k*-Means extended to partial distance (*MWk*-Means*PD*), and intelligent Minkwoski Weighted *k*-Means extended to partial distance (*iMWk*-Means*PD*) which are extension of their *k*-Means variants so that, the partial distance variants can handle missing values without the need of any imputation method.

ii) Two feature weighting based feature selection methods, namely feature selection using mean feature weight (*mean*-FSFW), and feature selection using maximum feature weight (*max*-FSFW), which are effective in reducing feature space cardinality while preserving maximum cluster structure.

iii) A method to convert data from p_{old}-norm to p_{new}-norm, which allows an analyst to transform a synthetic or real-world dataset which are normally only available in 2-norm.

iv) The extension of the proposed feature selection algorithms: *mean*-FSFWext*PD* and *max*-FSFWext*PD* where the partial-distance is used to address the missing values.

9.2 Observations

The following observations have been made:

i) Projection of clusters in p-norms for $p < 2$ separates them further and hence improves the performance of *k*-Means types algorithm and vice versa as p in p-norms increases.

ii) Minkowski variants of *k*-Means are able to detect the shape of clusters and are equally effective for cluster analysis of datasets, defined in p-norms other than 2-norm.

iii) A sharp drop in performance of weighted Minkwoski variants of *k*-Means is observed when a dataset is defined in 1-norm and distance coefficient of Minkowski metric is 1.

iv) Performance of weighted variant of *k*-Means is better when cluster based regression methods are used for imputation of missing values in synthetic and real-world (UCI) datasets when including noise in the dataset.

v) Partial distance approach is better in weighted variant of *k*-Means for Gaussian mixed model.

vi) Feature weighting can be effectively used for feature selection especially, in presence of noise.

vii) Feature selection using mean feature weights (*mean*-FSFW) has better cluster recovery for Gaussian mixed model whereas, in real-world (UCI) dataset feature selection using maximum feature weights (*max*-FSFW) is better choice.

viii) *max*-FSFW is slightly better than *mean*-FSFW for cluster recovery whereas *mean*-FSFW is better for reducing feature space cardinality.

ix) Both *mean*-FSFW and *max*-FSFW are effective in feature selection for larger dataset as well.

x) Both of the proposed feature selection algorithms retain the features required by the classification algorithms in the majority of tested datasets.

xi) *max*-FSFW with partial distance (*max*-FSFW*PD*) produced better results compared to other variants of FSFW with or without partial distance.

9.3 Limitations

The algorithm for transforming data from p_{old} norm to p_{new} norm has a precision problem when p tends to infinity, which is visible when displaying results. This algorithm works well for p less than 100. When the value of p moves towards infinity ($p >= 100$) two cases of misbehaviour are identified. First, some data points move towards infinity and cannot be reversed back to original

p-norm. Second, some data points move closer to centre while reverting the data points to their original p-norms.

The proposed feature selection algorithms, *mean*-FSFS and *max*-FSFS, use Minkowski weighted *k*-means with modified partial distance metric. *MWk*-Means used Minkowski center based on conversion of a convex function. The partial distance approach is yet to be extended to the conversion of a convex function, which limits using one of the imputation methods for the conversion.

The use of a partial distance approach to Minkowksi metric for cluster analysis has been extended. However, evaluation of these clusters is still not fully unsupervised as little work has been done so far on cluster validation index capable to work with missing values implicitly.
Feature weighting in the proposed feature selection algorithms assigns weights to the features based on their relevance. Theoretically, redundant (correlated) features are likely to have similar weights and therefore do not contribute towards selection.

9.4 Further research

Subspace feature selection is one of the emerging techniques in data mining. The proposed algorithms use feature weights based on clusters. This concept can be further extended to subspace feature selection which can be more beneficial when information is sparsely distributed.

The partial distance variants of the proposed feature selection methods can be further extended to unsupervised feature selection by enriching the available cluster validation index to deal with missing values without imputation methods. For this, the similarity metric used in the cluster validation index can be modified with partial distance.

The feature similarity index is successful in identifying redundant (correlated) features [143]. The inclusion of the feature similarity index in the proposed feature weighting based feature selection algorithm can make the algorithm more effective at identifying and removing redundant (correlated) features with further work.

The proposed feature selection algorithms already deal with feature weighting for cluster analysis. The partial distance approach can benefit from this weight and can be further extended to weighted partial distance.

Other areas of further research include extension of these concepts, both imputation of missing values and feature selection on constraint based clustering or semi-supervised clustering. The feature weighting-based feature selection approach can be equally important in other clustering approaches like density-based, hierarchical and spectral clustering. In fact, density-based clustering will be more effective with sub-space feature selection using feature weighting and is worth exploring. Similarly, the concept of using partial distance can add a new dimension to deal with missing values without 'false' imputations in clustering algorithms.

References

[1] Jennifer G Dy and Carla E Brodley. Feature selection for unsupervised learning. *Journal of machine learning research*, 5(Aug):845–889, 2004.

[2] Deepak Panday, Renato Cordeiro de Amorim, and Peter Lane. Feature weighting as a tool for unsupervised feature selection. *Information processing letters*, 129:44–52, 2018.

[3] Deepak Panday, Peter Lane, and Na Halian. Using feature weighting as a tool for clustering application. *Proceedings of Abstracts Engineering and Computer Science Research Conference*, pages 47–49, 2019.

[4] Boris Mirkin. *Clustering: a data recovery approach*. CRC Press, 2012.

[5] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[6] Douglas Steinley. K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, 2006.

[7] Peter HA Sneath, Robert R Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* University of California Press, 1973.

[8] KJ Holzinger and HH Harman. Factor analysis. chicago: Univer, 1941.

[9] Richard O Duda and Peter E Hart. Pattern recognition and scene analysis, 1973.

[10] Joshua Zhexue Huang, Jun Xu, Michael Ng, and Yunming Ye. Weighting method for feature selection in k-means. *Computational Methods of feature selection*, pages 193–209, 2008.

[11] Boris Mirkin. *Clustering For Data Mining: A Data Recovery Approach (Chapman & Hall/Crc Computer Science)*. Chapman & Hall/CRC, 2005.

[12] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[13] Arindam Banerjee and Joydeep Ghosh. Clickstream clustering using weighted longest common subsequences. In *Proceedings of the web mining workshop at the 1st SIAM conference on data mining*, volume 143, page 144. Citeseer, 2001.

[14] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[15] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[16] Harold Edson Driver and Alfred Louis Kroeber. *Quantitative expression of cultural relationships*, volume 31. University of California Press, 1932.

[17] Boris Mirkin. *Mathematical classification and clustering*, volume 11. Springer Science & Business Media, 1996.

[18] ID Mandel. Clustering analysis. *Finansy i Statistica, Moscow*, 1988.

[19] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[20] Geoffrey H Ball and David J Hall. A clustering technique for summarizing multivariate data. *Behavioral science*, 12(2):153–155, 1967.

[21] Fionn Murtagh and Pierre Legendre. WardâĂŹs hierarchical agglomerative clustering method: which algorithms implement wardâĂŹs criterion? *Journal of classification*, 31(3):274–295, 2014.

[22] Abdellah Idrissi, Hajar Rehioui, Abdelquoddouss Laghrissi, and Sara Retal. An improvement of denclue algorithm for the data clustering. In *2015 5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, pages 1–6. IEEE, 2015.

[23] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[24] Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2010.

[25] Renato Cordeiro de Amorim and Boris Mirkin. A clustering-based approach to reduce feature redundancy. In *Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions*, pages 465–475. Springer, 2016.

[26] Richard M Cormack. A review of classification. *Journal of the Royal Statistical Society: Series A (General)*, 134(3):321–353, 1971.

[27] John A Hartigan. Clustering algorithms. *xxx*, 1975.

[28] Walter D Fisher. On grouping for maximum homogeneity. *Journal of the American statistical Association*, 53(284):789–798, 1958.

[29] Douglas R Cox. Note on grouping. *Journal of the American Statistical Association*, 52(280):543–547, 1957.

[30] Michael R Anderberg. *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*, volume 19. Academic press, 2014.

[31] James M Lattin, J Douglas Carroll, and Paul E Green. *Analyzing multivariate data*. Thomson Brooks/Cole Pacific Grove, CA, 2003.

[32] John A Hartigan. A k-means clustering algorithm: Algorithm as 136. *Appl. Stat.*, 28:126–130, 1979.

[33] M Goyal and S Kumar. Improving the initial centroids of k-means clustering algorithm to generalize its applicability. *Journal of The Institution of Engineers (India): Series B*, 95(4):345–350, 2014.

[34] José M Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[35] Emanuel Falkenauer and Arnaud Marchand. Using k-means? consider arrayminer. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, Las Vegas*, 2001.

[36] Douglas Steinley. Local optima in k-means clustering: what you don't know may hurt you. *Psychological methods*, 8(3):294, 2003.

[37] Manoj Kumar Gupta and Pravin Chandra. Pk-means: k-means using partition based cluster initialization method. In *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*, 2019.

[38] MM Astrahan. Speech analysis by clustering, or the hyperphoneme method. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1970.

[39] Phipps Arabie and Lawrence J Hubert. Combinatorial data analysis. *Annual review of psychology*, 43(1):169–203, 1992.

[40] Glenn W Milligan and Lisa M Sokol. A two-stage clustering algorithm with robust recovery characteristics. *Educational and psychological measurement*, 40(3):755–759, 1980.

[41] Glenn W Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *psychometrika*, 45(3):325–342, 1980.

[42] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[43] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schroedl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[44] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[45] Silvio Lattanzi and Christian Sohler. A better k-means++ algorithm via local search. In *International Conference on Machine Learning*, pages 3662–3671. PMLR, 2019.

[46] Parimarjan Negi, Prafull Sharma, Vivek Jain, and Bahman Bahmani. K-means++ vs. behavioral biometrics: One loop to rule them all. In *NDSS*, 2018.

[47] Jan YK Chan and Alex Po Leung. Efficient k-means++ with random projection. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 94–100. IEEE, 2017.

[48] Pengfei Zhu, Wangmeng Zuo, Lei Zhang, Qinghua Hu, and Simon CK Shiu. Unsupervised feature selection by regularized self-representation. *Pattern Recognition*, 48(2):438–446, 2015.

[49] Edgar Chávez and Gonzalo Navarro. Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces. *Information Processing Letters*, 85(1):39–46, 2003.

[50] Tatjana Pavlenko. On feature selection, curse-of-dimensionality and error probability in discriminant analysis. *Journal of Statistical Planning and Inference*, 115(2):565–584, 2003.

[51] Vladimir Pestov. On the geometry of similarity search: dimensionality curse and concentration of measure. *Information Processing Letters*, 73(1):47–51, 2000.

[52] Jean-Charles Lamirel, Pascal Cuxac, and Kafil Hajlaoui. A novel approach to feature selection based on quality estimation metrics. In *Advances in Knowledge Discovery and Management*, pages 121–140. Springer, 2017.

[53] Jean-Charles Lamirel, Ingrid Falk, and Claire Gardent. Federating clustering and cluster labelling capabilities with a single approach based on feature maximization: French verb classes identification with igngf neural clustering. *Neurocomputing*, 147:136–146, 2015.

[54] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[55] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. *FSDM*, 10:4–13, 2010.

[56] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[57] Jennifer G Dy. Unsupervised feature selection. *Computational methods of feature selection*, pages 19–39, 2008.

[58] Elad Yom-Tov and Gideon F Inbar. Feature selection for the classification of movements from single movement-related potentials. *IEEE Transactions on neural systems and rehabilitation engineering*, 10(3):170–177, 2002.

[59] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125, 1994.

[60] Michael E Farmer, Shweta Bapna, and Anil K Jain. Large scale feature selection using modified random mutation hill climbing. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 287–290. IEEE, 2004.

[61] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[62] Edward B Fowlkes, Ram Gnanadesikan, and John R Kettenring. Variable selection in clustering. *Journal of classification*, 5(2):205–228, 1988.

[63] Frank J Carmone Jr, Ali Kara, and Sarah Maxwell. Hinov: A new model to improve market segment definition by identifying noisy variables. *Journal of Marketing Research*, 36(4):501–509, 1999.

[64] Manoranjan Dash and Huan Liu. Handling large unsupervised data via dimensionality reduction. In *1999 ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, 1999.

[65] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[66] Nicolas Vandenbroucke, Ludovic Macaire, and J-G Postaire. Unsupervised color texture feature extraction and selection for soccer image segmentation. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 2, pages 800–803. IEEE, 2000.

[67] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[68] David L Donoho and Yaakov Tsaig. Fast solution of l_1 norm minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54(11):4789–4812, 2008.

[69] Pengfei Zhu, Wencheng Zhu, Weizhi Wang, Wangmeng Zuo, and Qinghua Hu. Non-convex regularized self-representation for unsupervised feature selection. *Image and Vision Computing*, 60:22–29, 2017.

[70] Yugen Yi, Wei Zhou, Yuanlong Cao, Qinghua Liu, and Jianzhong Wang. Unsupervised feature selection with graph regularized nonnegative self-representation. In *Chinese Conference on Biometric Recognition*, pages 591–599. Springer, 2016.

[71] Chang Tang, Xinwang Liu, Miaomiao Li, Pichao Wang, Jiajia Chen, Lizhe Wang, and Wanqing Li. Robust unsupervised feature selection via dual self-representation and manifold regularization. *Knowledge-Based Systems*, 145:109–120, 2018.

[72] Y. Gu, K. Li, Z. Guo, and Y. Wang. Semi-supervised k-means ddos detection method using hybrid feature selection algorithm. *IEEE Access*, 7:64351–64365, 2019.

[73] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004.

[74] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[75] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11, 2004.

[76] Xu-Kui Yang, Liang He, Dan Qu, and Wei-Qiang Zhang. Semi-supervised minimum redundancy maximum relevance feature selection for audio classification. *Multimedia Tools and Applications*, 77(1):713–739, 2018.

[77] Yintong Wang, Jiandong Wang, Hao Liao, and Haiyan Chen. An efficient semi-supervised representatives feature selection algorithm based on information theory. *Pattern Recognition*, 61:511–523, 2017.

[78] Khalid Benabdeslem and Mohammed Hindawi. Efficient semi-supervised feature selection: constraint, relevance, and redundancy. *IEEE transactions on knowledge and data engineering*, 26(5):1131–1143, 2013.

[79] Sandeep Yaramakala and Dimitris Margaritis. Speculative markov blanket discovery for optimal feature selection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[80] Dietrich Wettschereck, David W Aha, and Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1):273–314, 1997.

[81] Wayne S DeSarbo, J Douglas Carroll, Linda A Clark, and Paul E Green. Synthesized clustering: A method for amalgamating alternative clustering bases with differential weighting of variables. *Psychometrika*, 49(1):57–78, 1984.

[82] Geert De Soete. Ovwtre: A program for optimal variable weighting for ultrametric and additive tree fitting. *Journal of Classification*, 5(1):101–104, 1988.

[83] Geert De Soete. Optimal variable weighting for ultrametric and additive tree clustering. *Quality and Quantity*, 20(2-3):169–180, 1986.

[84] Wayne S DeSarbo and Vijay Mahajan. Constrained classification: the use of a priori information in cluster analysis. *Psychometrika*, 49(2):187–215, 1984.

[85] Dharmendra S Modha and W Scott Spangler. Feature weighting in k-means clustering. *Machine learning*, 52(3):217–237, 2003.

[86] Elaine Y Chan, Wai Ki Ching, Michael K Ng, and Joshua Z Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern recognition*, 37(5):943–952, 2004.

[87] Joshua Zhexue Huang, Michael K Ng, Hongqiang Rong, and Zichen Li. Automated variable weighting in k-means type clustering. *IEEE transactions on pattern analysis and machine intelligence*, 27(5):657–668, 2005.

[88] Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, 2005.

[89] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer, 2004.

[90] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[91] Liping Jing, Michael K Ng, and Joshua Zhexue Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on knowledge and data engineering*, 19(8):1026–1041, 2007.

[92] Renato Cordeiro De Amorim and Boris Mirkin. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition*, 45(3):1061–1075, 2012.

[93] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[94] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[95] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[96] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489, 1996.

[97] Ito Wasito and Boris Mirkin. Nearest neighbour approach in the least-squares data imputation algorithms. *Information Sciences*, 169(1-2):1–25, 2005.

[98] Alan C Acock. Working with missing values. *Journal of Marriage and family*, 67(4):1012–1028, 2005.

[99] Loai AbdAllah and Ilan Shimshoni. A distance function for data with missing values and its application. In *Proceedings of the 2013th International Conference on Data Mining and Knowledge Engineering*, 2013.

[100] Lorenzo Beretta and Alessandro Santaniello. Nearest neighbor imputation algorithms: a critical evaluation. *BMC medical informatics and decision making*, 16(3):197–208, 2016.

[101] Jiahua Chen and Jun Shao. Nearest neighbor imputation for survey data. *Journal of official statistics*, 16(2):113, 2000.

[102] Per Jonsson and Claes Wohlin. An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 108–118. IEEE, 2004.

[103] Wang Ling and Fu Dong-Mei. Estimation of missing values using a weighted k-nearest neighbors algorithm. In *2009 International Conference on Environmental Science and Information Application Technology*, volume 3, pages 660–663. IEEE, 2009.

[104] Emil Eirola, Gauthier Doquire, Michel Verleysen, and Amaury Lendasse. Distance estimation in numerical data sets with missing values. *Information Sciences*, 240:115–128, 2013.

[105] Jason Van Hulse and Taghi M Khoshgoftaar. Incomplete-case nearest neighbor imputation in software measurement data. *Information Sciences*, 259:596–610, 2014.

[106] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[107] Trond Hellem Bø, Bjarte Dysvik, and Inge Jonassen. Lsimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic acids research*, 32(3):e34–e34, 2004.

[108] Hyunsoo Kim, Gene H Golub, and Haesun Park. Missing value estimation for dna microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

[109] Ming Ouyang, William J Welsh, and Panos Georgopoulos. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20(6):917–923, 2004.

[110] Xiangchao Gan, Alan Wee-Chung Liew, and Hong Yan. Microarray missing data imputation based on a set theoretic framework and biological knowledge. *Nucleic Acids Research*, 34(5):1608–1619, 2006.

[111] Dankyu Yoon, Eun-Kyung Lee, and Taesung Park. Robust imputation method for missing values in microarray data. *BMC bioinformatics*, 8(2):1–7, 2007.

[112] Bankat M Patil, Ramesh C Joshi, and Durga Toshniwal. Missing value imputation based on k-mean clustering with weighted distance. In *International Conference on Contemporary Computing*, pages 600–609. Springer, 2010.

[113] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. Robust similarity measures for mobile object trajectories. In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pages 721–726. IEEE, 2002.

[114] Ankita Vimal, Satyanarayana R Valluri, and Kamalakar Karlapalem. An experiment with distance measures for clustering. In *COMAD*, pages 241–244. Citeseer, 2008.

[115] Jhansi Rani Vennam and Soujanya Vadapalli. Syndeca: A tool to generate synthetic datasets for evaluation of clustering algorithms. In *COMAD*, pages 27–36. Citeseer, 2005.

[116] Espn cricket information. `https://cricinfo.com`. Accessed: 2021-07-27.

[117] Endre Pap, Mirjana Štrboja, and Imre Rudas. Pseudo-lp space and convergence. *Fuzzy Sets and Systems*, 238:113–128, 2014.

[118] Suman K Bera, Deeparnab Chakrabarty, Nicolas J Flores, and Maryam Negahbani. Fair algorithms for clustering. *arXiv preprint arXiv:1901.02393*, 2019.

[119] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[120] Vincent Cohen-Addad and CS Karthik. Inapproximability of clustering in lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539. IEEE, 2019.

[121] Chen Fu and Jianhua Yang. Granular classification for imbalanced datasets: A minkowski distance-based method. *Algorithms*, 14(2):54, 2021.

[122] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 169–182, 2021.

[123] Yaling Pei and Osmar Zaïane. A synthetic data generator for clustering and outlier analysis. 2006.

[124] Yannis Theodoridis and Mario A Nascimento. Generating spatiotemporal datasets on the www. *ACM SIGMOD Record*, 29(3):39–43, 2000.

[125] Ibm quest synthetic data generator. `https://ibm-quest-synthetic-data-generator.soft112.com`. Accessed: 2021-07-09.

[126] Steve Kalke and W-D Richter. Simulation of the p-generalized gaussian distribution. *Journal of Statistical Computation and Simulation*, 83(4):641–667, 2013.

[127] Ismail Bin Mohamad and Dauda Usman. Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17):3299–3303, 2013.

[128] Glenn W Milligan and Martha C Cooper. A study of standardization of variables in cluster analysis. *Journal of classification*, 5(2):181–204, 1988.

[129] Catherine M Schaffer and Paul E Green. An empirical comparison of variable standardization methods in cluster analysis. *Multivariate Behavioral Research*, 31(2):149–167, 1996.

[130] Anne M Stoddard. Standardization of measures prior to cluster analysis. *Biometrics*, pages 765–773, 1979.

[131] Catherine Blake and Christopher J Merz. {UCI} repository of machine learning databases. 1998.

[132] Douglas Steinley. Standardizing variables in k-means clustering. In *Classification, clustering, and data mining applications*, pages 53–60. Springer, 2004.

[133] Kamarul Ismail, Nasir Nayan, and Siti Naielah Ibrahim. Improving the tool for analyzing malaysiaâĂŹs demographic change: Data standardization analysis to form geodemographics classification profiles using k-means algorithms. *Geografia-Malaysian Journal of Society and Space*, 12(6), 2017.

[134] Hichem Frigui and Olfa Nasraoui. Unsupervised learning of prototypes and attribute weights. *Pattern recognition*, 37(3):567–581, 2004.

[135] R. C. de Amorim. A survey on feature weighting based k-means algorithms. *Journal of Classification*, 33(2):210–242, 2016.

[136] Mark Ming-Tso Chiang and Boris Mirkin. Experiments for the number of clusters in k-means. In *Progress in Artificial Intelligence*, pages 395–405. Springer, 2007.

[137] Renato Cordeiro de Amorim. Constrained clustering with minkowski weighted k-means. In *2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 13–17. IEEE, 2012.

[138] Richard Dubes and Anil K Jain. Validity studies in clustering methodologies. *Pattern recognition*, 11(4):235–254, 1979.

[139] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[140] Katherine S Pollard and Mark J Van Der Laan. A method to identify significant clusters in gene expression data. 2002.

[141] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[142] Jorge M Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International Conference on Artificial Neural Networks*, pages 175–184. Springer, 2009.

[143] Renato Cordeiro de Amorim and Boris Mirkin. Removing redundant features via clustering: preliminary results in mental task separation. In *Proceedings of the 8th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, pages 7–9, 2012.

[144] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[145] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[146] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[147] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Series in Statistics. Springer, 2009.

[148] Andrew Clark, Qiqiang Hou, Linda Bushnell, and Radha Poovendran. Maximizing the smallest eigenvalue of a symmetric matrix: A submodular optimization approach. *Automatica*, 95:446–454, 2018.

[149] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[150] Shin-Mu Tseng, Kuo-Ho Wang, and Chien-I Lee. A pre-processing method to deal with missing values by integrating clustering and regression techniques. *Applied Artificial Intelligence*, 17(5-6):535–544, 2003.

[151] Ludmila Himmelspach and Stefan Conrad. Clustering approaches for data with missing values: Comparison and evaluation. In *Digital Information Management (ICDIM), 2010 Fifth International Conference on*, pages 19–28. IEEE, 2010.

[152] Renato Cordeiro de Amorim and Peter Komisarczuk. On initializations for the minkowski weighted k-means. In *International Symposium on Intelligent Data Analysis*, pages 45–55. Springer, 2012.

[153] Renato Cordeiro de Amorim and Christian Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126–145, 2015.

[154] Yu Xiao, Tieyong Zeng, Jian Yu, and Michael K Ng. Restoration of images corrupted by mixed gaussian-impulse noise via l1–l0 minimization. *Pattern Recognition*, 44(8):1708–1720, 2011.

[155] Yan-Jun Liu, Shao-Cheng Tong, and Wei Wang. Adaptive fuzzy output tracking control for a class of uncertain nonlinear systems. *Fuzzy Sets and Systems*, 160(19):2727–2754, 2009.

[156] Christian Clason. LâĹđ fitting for inverse problems with uniform noise. *Inverse Problems*, 28(10):104007, 2012.

[157] Anekal Sripad and Donald Snyder. A necessary and sufficient condition for quantization errors to be uniform and white. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(5):442–448, 1977.

[158] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

[159] John Ross Quinlan. The effect of noise on concept learning. *Machine learning: An artificial intelligence approach*, 2:149–166, 1986.

[160] Mauricio A Hernández and Salvatore J Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37, 1998.

[161] Mark Ming-Tso Chiang and Boris Mirkin. Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads. *Journal of classification*, 27(1):3–40, 2010.

[162] Renato Cordeiro de Amorim. Constrained clustering with minkowski weighted k-means. In *2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 13–17. IEEE, 2012.

[163] Siwei Wang, Miaomiao Li, Ning Hu, En Zhu, Jingtao Hu, Xinwang Liu, and Jianping Yin. K-means clustering with incomplete data. *IEEE Access*, 7:69162–69171, 2019.

[164] John K Dixon. Pattern recognition with partly missing data. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(10):617–621, 1979.

[165] Shounak Datta, Supritam Bhattacharjee, and Swagatam Das. Clustering with missing features: a penalized dissimilarity measure based approach. *Machine Learning*, 107(12):1987–2025, 2018.

[166] Ernie G Kalnins, W Miller Jr, and GS Pogosyan. Superintegrability and associated polynomial solutions: Euclidean space and the sphere in two dimensions. *Journal of Mathematical Physics*, 37(12):6439–6467, 1996.

[167] LE Blumenson. A derivation of n-dimensional spherical coordinates. *The American Mathematical Monthly*, 67(1):63–66, 1960.

[168] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

Appendix A

Coordinates System

A.1 Cartesian to polar conversion

Cartesian and polar are two popular coordinate systems on the Euclidean plane[166]. A polar coordinate in two dimensional Euclidean plane is represented by a radius (r), distance from a reference point (or pole) and an angle θ from a reference direction. The reference direction, a ray originated from the pole, is called the polar axis. In the Cartesian coordinate system, a point in the two dimensional Euclidean plane is represented by (x_1, x_2) where x_1 and x_2 are the displacement of the point from its origin along the two axes (x and y) respectively. The pole in polar coordinates is analogous to the origin of the Cartesian coordinate and the polar axes of the polar coordinate represents one of the Cartesian axes (usually x).

Conversion in 2D

Let a point P(r, θ_1) be a point in a polar coordinate, as shown in the figure below.

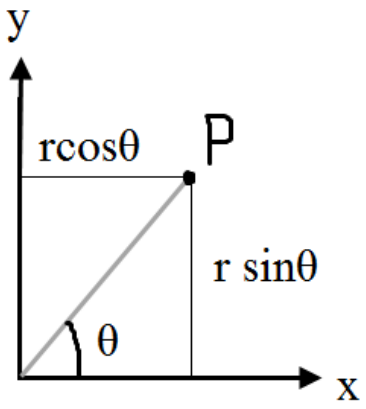


FIGURE A.1: Polar coordinate system in two dimensions.

The relationship between the two coordinate systems which is defined as:

$$\begin{aligned} x &= rcos\theta, r > 0 \\ y &= rsin\theta, 0 \geq \theta < 2\pi. \end{aligned} \tag{A.1}$$

where r is a radial coordinate range, θ is angular coordinate in polar coordinates, and x and y are two dimensions in Cartesian coordinates. The value of r in the equation A.1 is given by,

$$r = \sqrt{x^2 + y^2} \tag{A.2}$$

and the angular measurement(θ) is obtained by

$$\theta = acos\frac{x}{\sqrt{x^2 + y^2}} \tag{A.3}$$

Conversion in 3D
Let a point P(r, θ_1,θ_1) be a point in a polar coordinate, as shown in the figure below.

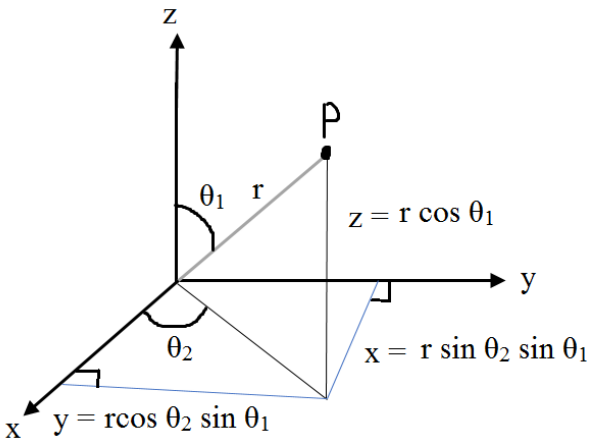


FIGURE A.2: Polar coordinate system in three dimensions.

$$\begin{aligned} z &= rcos\theta_1 \\ y &= rsin\theta_1 cos\theta_2 \\ x &= rsin\theta_1 sin\theta_2 \end{aligned} \tag{A.4}$$

where, (x, y, z) is a corresponding point in a Cartesian coordinate.
The radial coordinate r in the equation above is,

$$r = \sqrt{x^2 + y^2 + z^2} \tag{A.5}$$

and the angular coordinates θ_1, θ_2 are,

$$\begin{aligned} \theta_1 &= acos\frac{z}{\sqrt{x^2 + y^2 + z^2}} \\ \theta_2 &= acos\frac{y}{\sqrt{x^2 + y^2 + z^2}} \end{aligned} \tag{A.6}$$

Generalization,

Let a point P(r, θ_1, θ_1, ..., θ_m) be a point in a polar coordinate. Then its equivalent Cartesian coordinate can be derived as [167]:

$$\begin{aligned} x_1 &= rcos\theta_1 \\ x_2 &= rsin\theta_1 cos\theta_2 \\ x_3 &= rsin\theta_1 sin\theta_2 \\ &. \\ &. \\ &. \\ x_{m-1} &= rsin\theta_1 sin\theta_2 ....sin\theta_{m-2} cos\theta_{m-1} \\ x_m &= rsin\theta_1 sin\theta_2 ....sin\theta_{m-2} sin\theta_{m-1} \end{aligned} \tag{A.7}$$

where, $(x_1, x_2, ..., x_m)$ is a corresponding point in a Cartesian coordinate.
The radial coordinate r in the equation above is,

$$r = \sqrt{x_1^2 + x_2^2 + ... + x_m^2} \tag{A.8}$$

and the angular coordinates $\theta_1, \theta_2, ..., \theta_{m-1}$ are,

$$\begin{aligned}
\theta_1 &= acos \frac{x_1}{\sqrt{x_m^2 + x_{m-1}^2 + ... + x_1^2}} \\
\theta_2 &= acos \frac{x_2}{\sqrt{x_m^2 + x_{m-1}^2 + ... + x_2^2}} \\
&. \\
&. \\
&. \\
\theta_{m-2} &= acos \frac{x_2}{\sqrt{x_m^2 + x_{m-1}^2 + x_{m-2}^2}} \\
\theta_{m-1} &= \begin{cases} acos \frac{x_{m-1}}{\sqrt{x_m^2 + x_{m-1}^2}}, & x_m \geq 0 \\ 2\pi - acos \frac{x_{m-1}}{\sqrt{x_m^2 + x_{m-1}^2}} \end{cases}
\end{aligned} \tag{A.9}$$

Appendix B

Gaussian Distribution

The Gaussian distribution, also known as a normal distribution is a widely used model to simulate the distribution of continuous variables. The probability density function (PDF) of a Gaussian distribution x with mean μ and variance σ^2 is given by,

$$P(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{B.1}$$

B.0.1 Standard Gaussian Distribution

A standard G distribution is a special case of a Gaussian distribution with mean μ = 0 and variance σ^2=1 with a bell-shaped PDF which is illustrated in figure B.1.

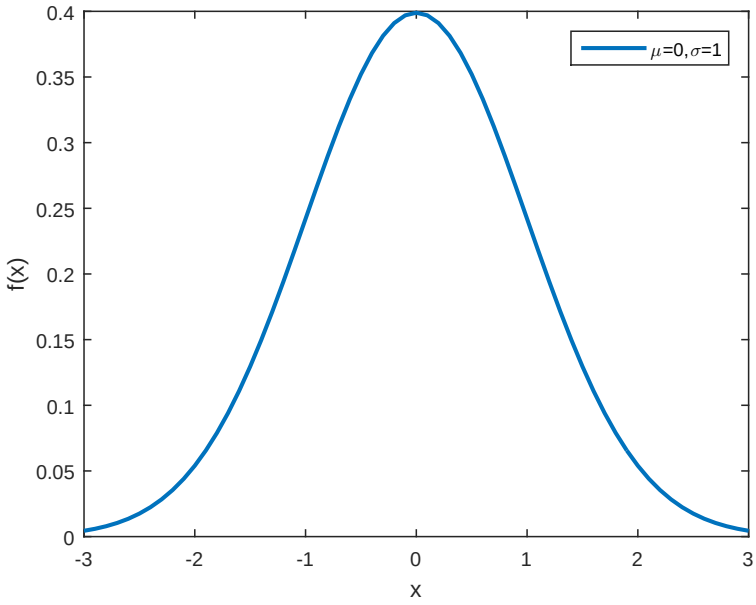


FIGURE B.1: Probability density function (PDF) of Gaussian Model with mean = 0 and sigma (standard deviation) = 1.

The figure B.1 above has a random number generated between (-∞,∞) along the x-axis such the mean of the numbers is equal to 3 and the standard deviation of the number is equal to 1. Y-axis in the graph is the probability density function (PDF) of the points. Basically, the graph in the figure represents PDF of a Gaussian distribution with mean 3 and standard deviation equal to 1. This is the standard case of Gaussian distribution also called as a normal distribution.

B.0.2 Gaussian Distribution in two Dimension

In two dimensions, each of the Gaussian PDF is an area covered under oval-shaped controlled lines along the two-dimensional planes.

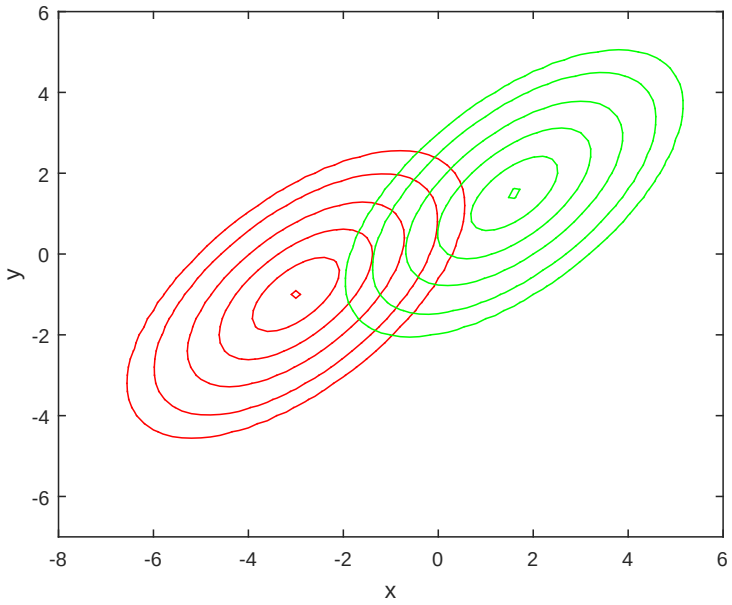


FIGURE B.2: Probability density functions (PDFs) of two Gaussians with two variate random variables x and y.

In the figure C.1, the red and green oval-shaped curve lines denote the control line of two Gaussian distributions with different mean and variance.

B.0.3 Generalization of Gaussian Distribution

For a D-dimensional vector x, the PDF of a multivariate Gaussian distribution is given by:

$$P(x|\mu,\Sigma) = \frac{1}{(2\Pi)^{D/2}\|\Sigma\|^{1/2}} e^{-1/2(x-\mu)^T\Sigma^{\frac{1}{x-\mu}}} \tag{B.2}$$

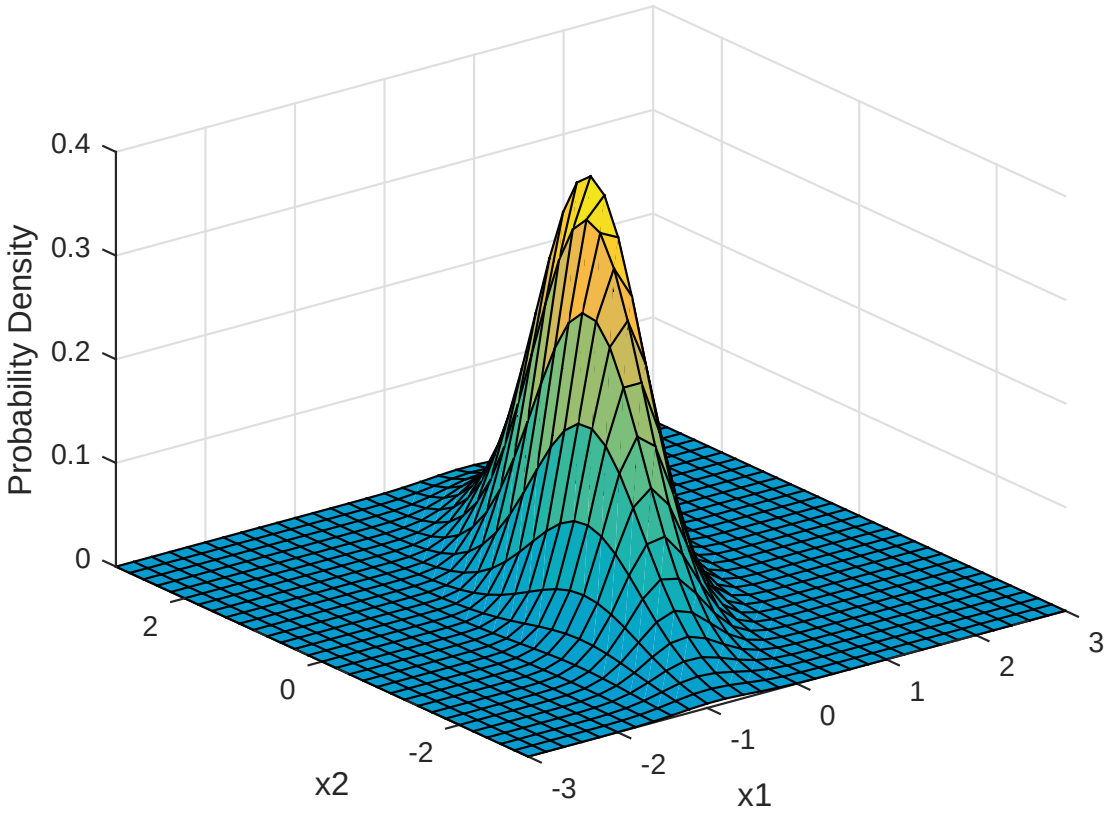


FIGURE B.3: Normalized Gaussian distribution in three dimension.

Appendix C

Gaussian Mixed Model

In the real-world datasets normally have more than one Gaussian component. Each of these components has its own probability density function. In terms of cluster analysis, each of these components can represents a cluster.

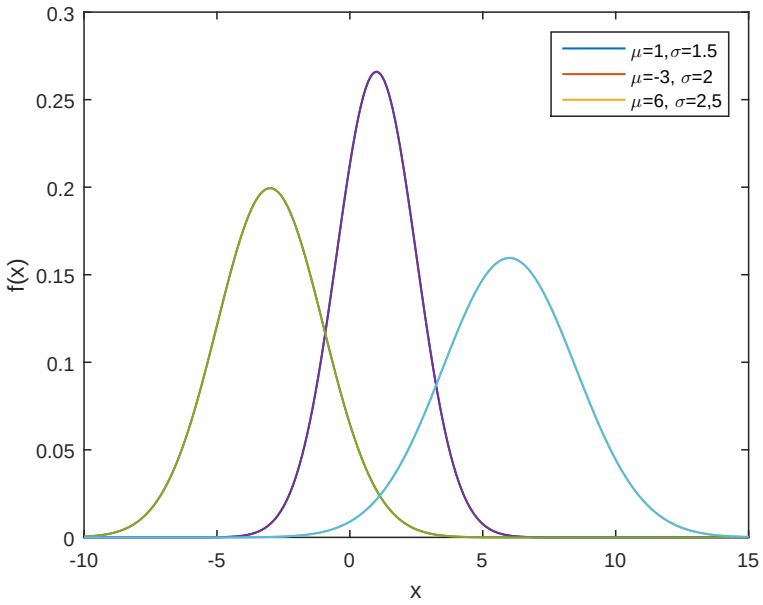


FIGURE C.1: Probability density functions (PDFs) of three univariate Gaussian components

The figure C.1 has three Gaussian components with different means and standard deviations.

C.0.1 Gaussian Mixed Model in one Dimension

Figure C.1 shows a dataset with three Gaussian components (clusters): the first component (green in the figure above) has a bell-shape PDF, the second component (purple in the figure above) has a stiff shape PDF and the last one (blue in the figure above) has a flat shape PDF.

The probability density function of the mixture of the three Gaussian components is given by a linear combination of the individual PDFs. Each peak from the individual distribution may create a local peak (maxima) in the mixed distribution. The PDF of the mixed Gaussian looks like ranges of mountains which is shown in as dotted line in the figure C.1.

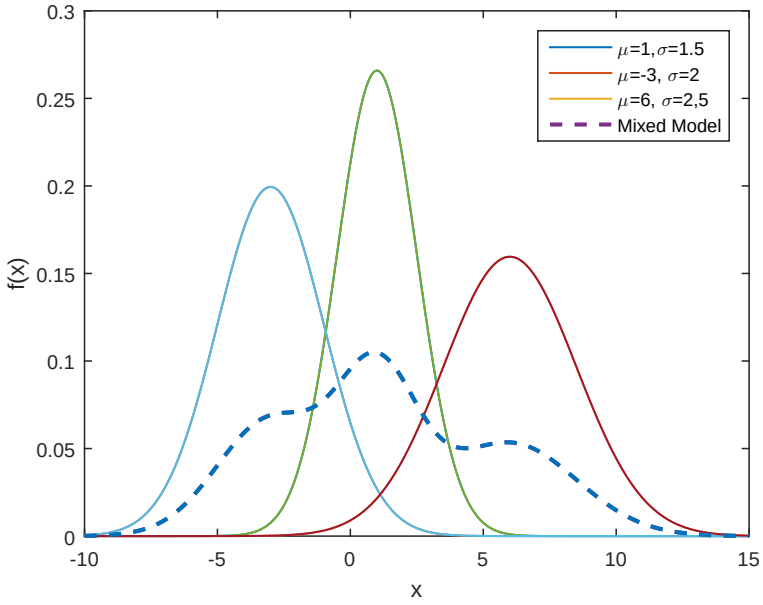


FIGURE C.2: Mixed model Gaussian (GMM) distribution of three univariate Gaussian components is given by a linear combination of the three Gaussians.

The figure C.2 combines the three Gaussian components in figure C.1. The mixed Gaussian is represented by dot line in the figure.

C.0.2 Gaussian Mixed Model (GMM) for two Dimensions

Control lines for the Gaussian mixture of the two Gaussian components are obtained by a linear combination of the control lines from each component. If we visualize each of these Gaussian components as a mountain, the mixed Gaussian is the range of mountains that is the topology of the mountains.

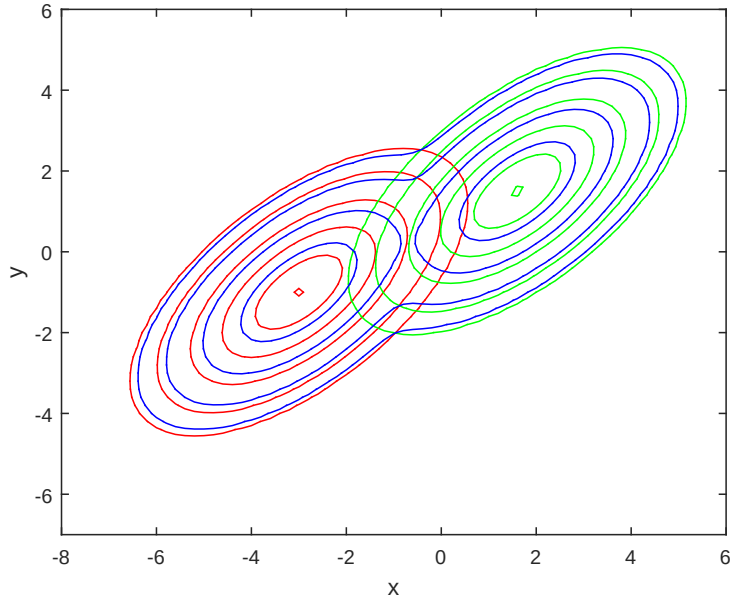


FIGURE C.3: Probability density functions (PDF) of two variate Gaussian components

The figure C.3 shows the linear combination of two Gaussian components defined in the figure B.2. The control lines (curves) in red and green color represents the control lines for each of the Gaussian whereas the control lines (cures) with blue color is the control lines formed by the linear combination of two Gaussians.

Appendix D

GMM Control Parameters and *k*

D.1 Control Parameters in GMM

The distribution of a dataset generated from GMM depends on several factors including feature space cardinality, number of clusters, location of centroid, shape of cluster and covariance matrix. Therefore,it is worth observing how these parameters affect the formation of clusters and cluster recovery. In this section, we discuss how feature space cardinality, number of Gaussian components and covariance matrix of Gaussian components effect on cluster recovery in *k*-Means. To evaluate the cluster recovery RAND index [[168]] is used.

For simplicity, we have used a spherical shaped Gaussian mixed model (GMM). This model requires a single variance parameter of each component. Centroid for each component is drawn randomly from a uniform distribution defined over the range 0 to 1. We vary the variance of GMM components in the range starting from .1 to 2, with an interval of 0.1. Also, feature space cardinality and the number of Gaussian component are drawn from 2,4,8,16,32, 64 and 2,4,8,16,32,64 respectively.

D.1.1 Visualization of GMM Datasets

As a visualization of a dataset provides an easy way to grasp patterns present in the dataset, we go through the visual inspection of the datasets for all possible combinations under our control environments defined above. We carried out principal component analysis (PCA) of each dataset and used the first two components to display the data points in the two-dimensional plane.

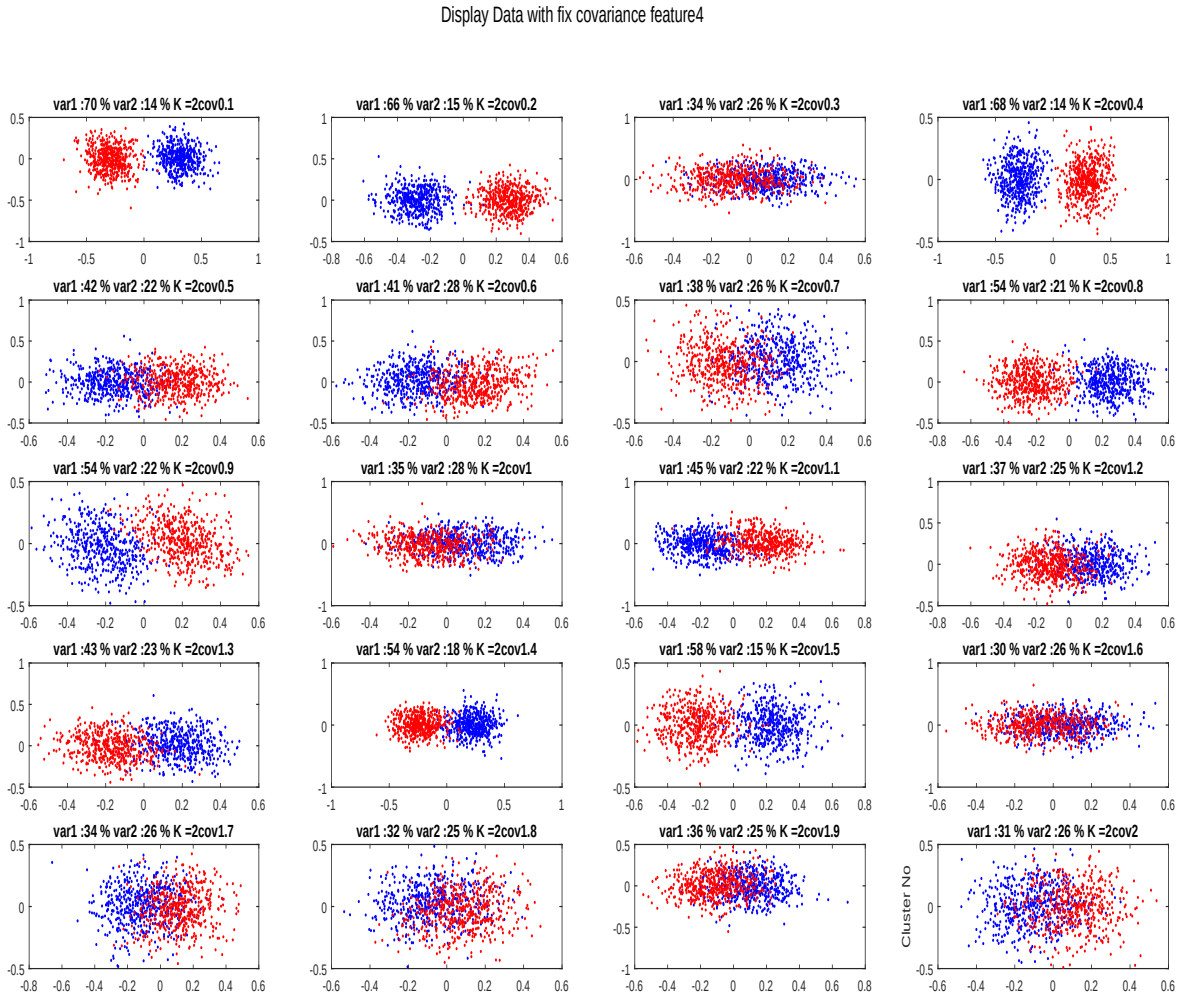


FIGURE D.1: Visualization of the synthetic datasets generated by Gaussian mixed models. These models consist of two GM components and have four variates (features). First two PCA)Principal component analysis) component are used to display the data points.

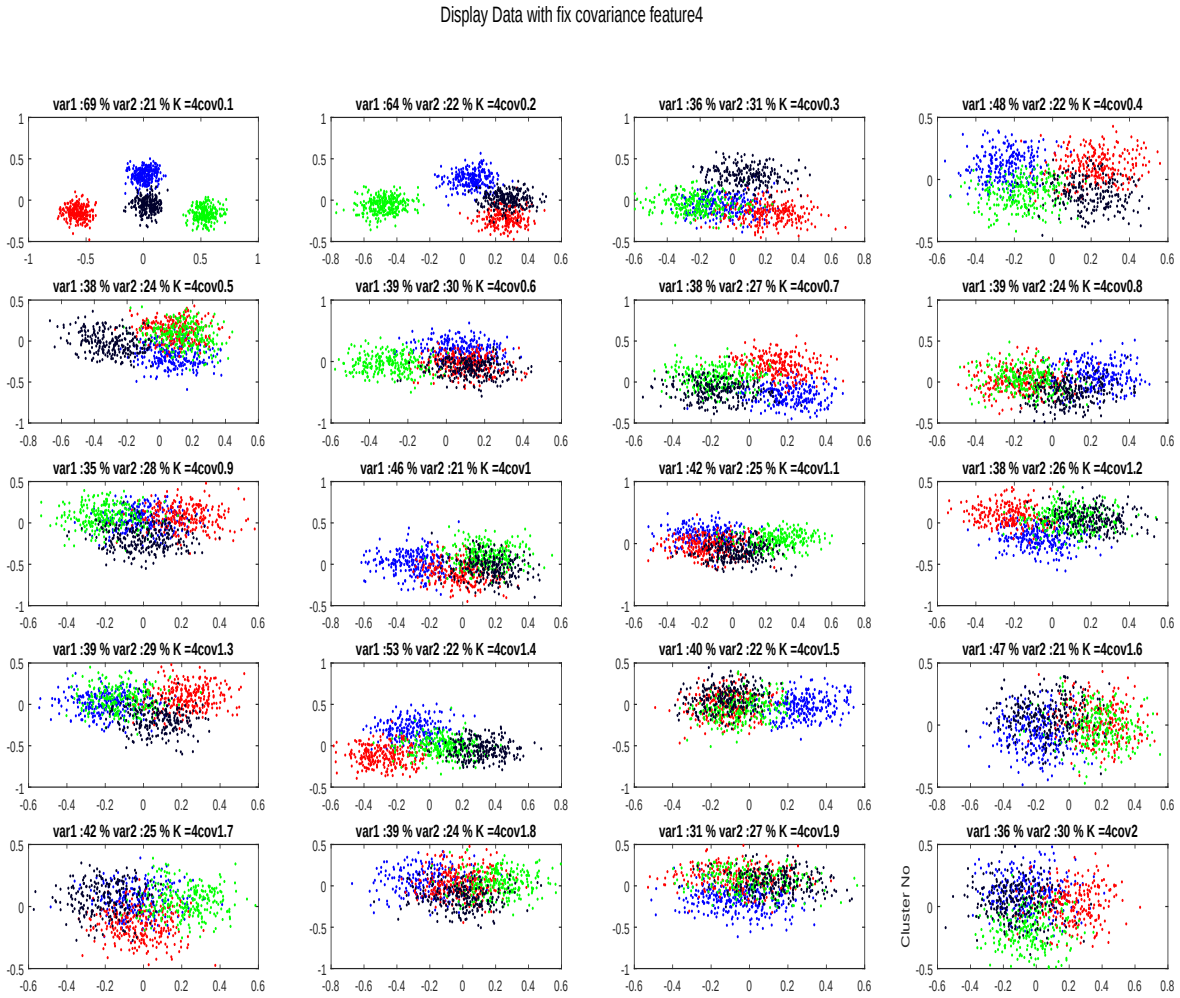
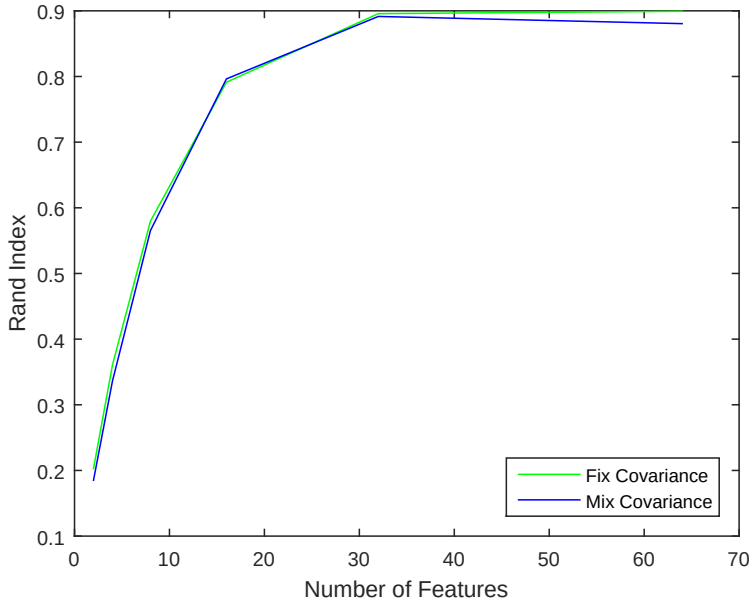


FIGURE D.2: Visualization of the synthetic datasets generated by spherical shaped Gaussian mixed models. These models consist of four Gaussian components and have four variates (features). First two components of PCA are used to display data points.

In the two figures above, we used different colors to distinguish different components of GMMs. feature space cardinality of Gaussian is set to four in both examples. But, in the first figure, the number of Gaussian components is set to two and in the second figure, the number of Gaussian components is set to four. The variance of the GMM models varies from 0.1 to 2 in an interval of 0.1 in each subplot. Each of the sub-plots displays variances of the first two PCA components, the number of Gaussian components and covariance of the model.

From data visualization in Figure D.1 and D.2, we observe that with the increase in the number of Gaussian components, overlapping between the clusters increases. Also, with the increase in covariance of GMM models, the data points are widely scattered get a higher resulting high chance of data overlapping.



[H]

FIGURE D.3: Performance of *k*-Means is influenced by the number of features of the Gaussian components.

D.1.2 Observation based on cluster recovery

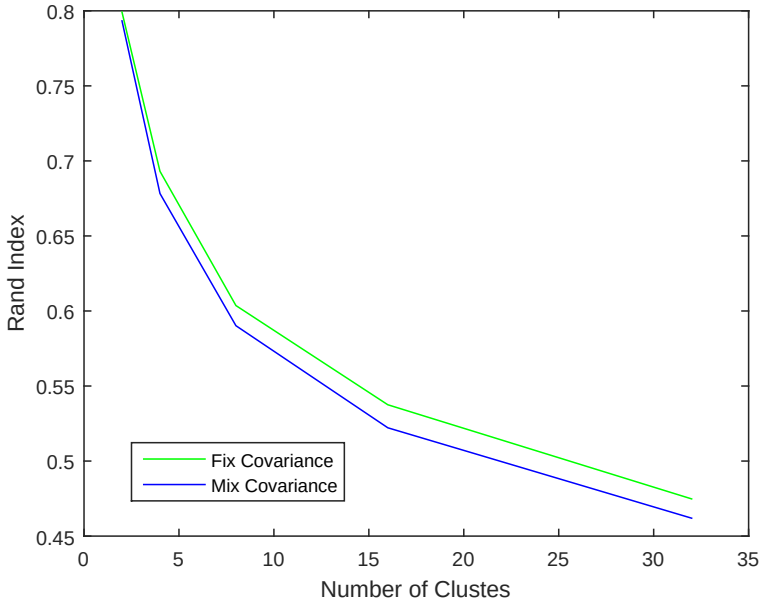
To choose the best parameter setting for GMM, we examine the performance of the *k*-Means on basis of cluster recovery (RAND index) for each of the combination. Again, for each combination, we compare the results with a Gaussian model drawn with the same configuration but only varying the covariance in the interval of +/-0.2.

- Number of feature

  With the increase in feature space, the performance of *k*-Means is improved in both cases: fixed covariance and mixed covariance.

- Number of clusters

  Again, with the increase in the number of Gaussian components, the performance of *k*-Means decreases in both cases. More important the tendency of cluster recovery remains the same in either case.

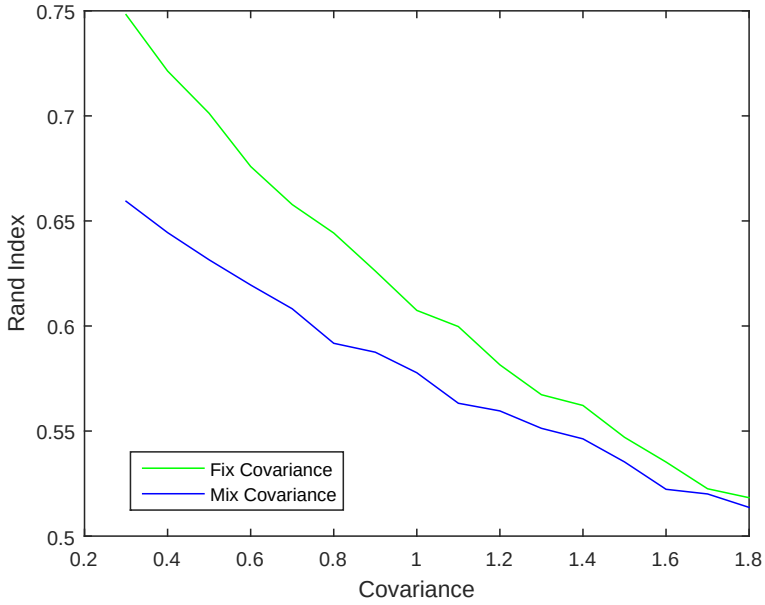


[H]

FIGURE D.4: Performance of *k*-Means against the number of Gaussian components.

- Affect of covariance

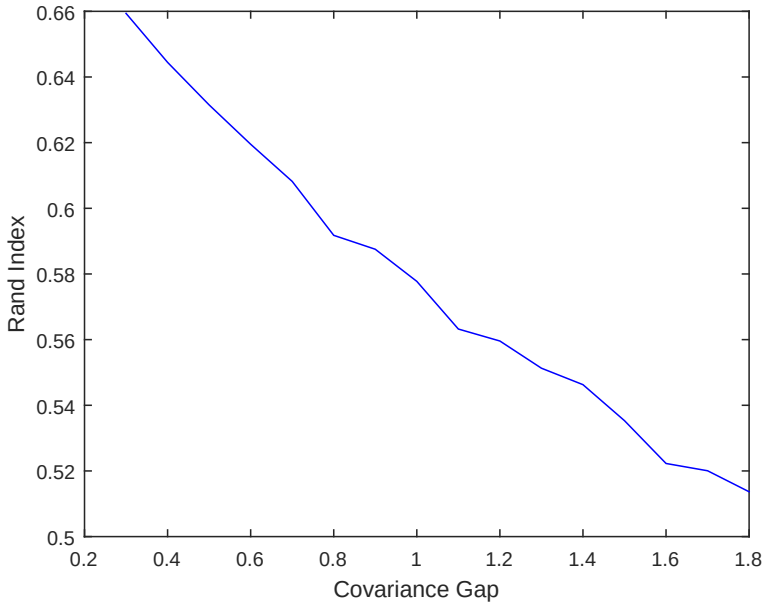
  Similar tend is obtained when we vary covariance. In overall, the increase in covariance had decreased cluster recovery of *k*-Means but the trend of cluster recovery remains the same in both cases of fix or interval covariance.



[H]

FIGURE D.5: Performance of *k*-Means against the covariance of the Gaussian components.

- Covariance interval



captionPerformance of *k*-Means against the range of covariance in Gaussian components with mixed covariance matrix.

We also notice that the increase in the window size of covariance in the second experiment decreases the performance of *k*-Means in terms of cluster recovery.

Appendix E

Effect of Noise in *k*-means

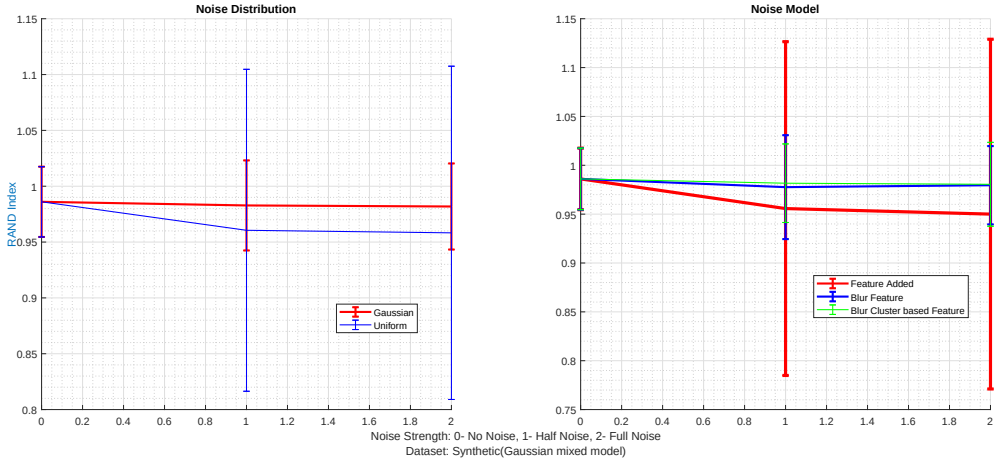


FIGURE E.1: Effect of noise on the performance of *k*-Means. Datasets are generated using Gaussian mixed models. Noise i generated from two different distributions: Gaussian and uniform distributions.

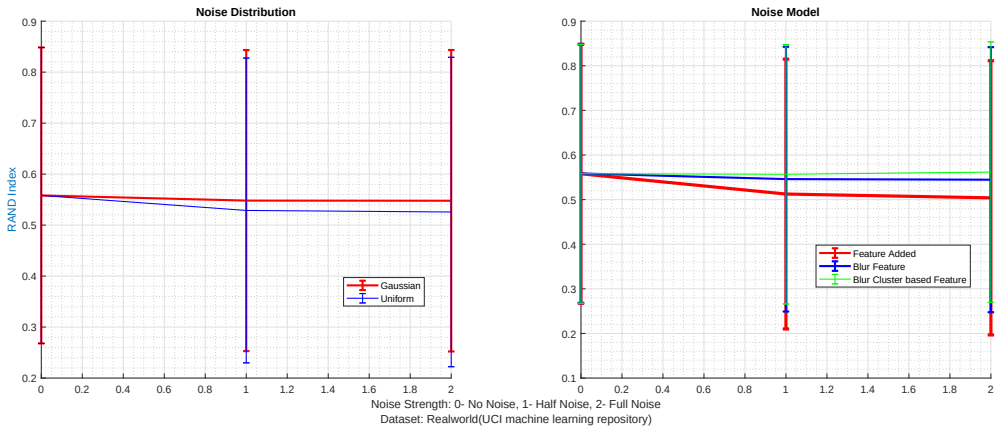


FIGURE E.2: Effect of noise on the performance of *k*-Means in real-world datasets. Datasets are generated using Gaussian mixed models. Noise are generated from two different distributions: Gaussian and uniform distributions.

From our experiments, we observe that uniform feature noise has greater influence in the performance of *k*-Means. However, in the real-world datasets feature noise from both the distributions have a similar effect on the performance of *k*-Means.

E.1 Using Cluster Validation Index in L_p

The table E.1 shows cluster validation index can be used to find appropriate distance coefficient for Minkowski metric in L_p space.

TABLE E.1: Performance of kMeans variants in different L_p. The Performance column contains mean/std .

	Cluser Recovery (ARI)							
			MWk-Means			*iMWk*-Means		
	k-Means	*ik*-Means	Silhoutee	Silhoutee(city block)	CH	Silhoutee	Silhoutee(city block)	CH
No Noise								
0.5	0.98584/0.031518	0.9857/0.031603	0.98649/0.031184	0.98619/0.031896	**0.98662**/0.031557	0.98322/0.042015	0.98279/0.042691	0.9697/0.061426
1	0.98593/0.031241	0.98605/0.031236	0.9863/0.031499	0.98627/0.032798	**0.98666**/0.031524	0.98364/0.041314	0.98357/0.041936	0.96872/0.06037
1.5	0.98585/0.031372	0.98591/0.031323	0.98644/0.031049	**0.98666**/0.031072	0.98657/0.030971	0.98343/0.041818	0.98336/0.041889	0.96545/0.066853
2	0.98571/0.031671	0.98584/0.031506	0.98663/0.03101	**0.98664**/0.031093	0.9866/0.031256	0.9834/0.041536	0.98347/0.04156	0.96436/0.072462
2.5	0.98578/0.031502	0.98583/0.031484	0.98592/0.03055	**0.98649**/0.0305	0.98618/0.031158	0.98352/0.04123	0.98367/0.041244	0.96774/0.071415
3	0.98582/0.031504	0.98583/0.031484	0.98644/0.030736	0.98663/0.030898	**0.98674**/0.030839	0.98086/0.047257	0.98103/0.047352	0.96924/0.062979
3.5	0.98578/0.031467	0.98579/0.03147	0.98676/0.02984	0.98689/0.029885	**0.98702**/0.029748	0.98101/0.047035	0.98107/0.047105	0.96607/0.066985
4	0.9857/0.03145	0.98579/0.031476	0.98662/0.030113	0.98242/0.065028	**0.98684**/0.030131	0.98416/0.038095	0.98429/0.038171	0.96793/0.06969
4.5	0.9857/0.031267	0.98576/0.031445	0.98669/0.0297	0.98701/0.029715	**0.9871**/0.029806	0.98407/0.038177	0.98422/0.03818	0.96757/0.069875
5	0.98567/0.031428	0.98576/0.031407	0.9869/0.029651	0.98712/0.02963	**0.98738**/0.029373	0.98403/0.038388	0.98421/0.038392	0.96962/0.061892
Half Noise								
0.5	0.72466/0.39393	0.7138/0.39986	0.85618/0.32193	0.94774/0.17081	0.83395/0.3373	0.87048/0.31698	**0.95157**/0.17119	0.53741/0.33594
1	0.86682/0.29942	0.81427/0.35008	0.94685/0.19606	**0.97695**/0.067162	0.9472/0.19607	0.92669/0.24259	0.97666/0.077366	0.60954/0.32627
1.5	0.92924/0.2286	0.90944/0.24958	0.92023/0.23264	0.97102/0.081616	0.92329/0.22966	0.94125/0.20001	**0.97249**/0.077358	0.67422/0.31756
2	0.92982/0.22659	0.93486/0.21093	0.95232/0.17196	**0.97918**/0.066469	0.94099/0.20241	0.9451/0.19448	0.97759/0.058043	0.70451/0.31296
2.5	0.96332/0.1361	0.92812/0.22559	0.9586/0.13942	**0.9756**/0.073384	0.96329/0.1353	0.96312/0.13296	0.97457/0.077209	0.74244/0.31337
3	0.96436/0.13443	0.92247/0.22698	0.9565/0.14226	0.97362/0.074372	0.9672/0.12633	0.96455/0.12926	**0.97564**/0.070878	0.76081/0.29288
3.5	0.96396/0.13515	0.95822/0.16373	0.95877/0.14006	**0.97642**/0.074976	0.96381/0.13514	0.95776/0.1365	0.971/0.084991	0.75983/0.29596
4	0.96351/0.13501	0.96535/0.12793	0.96229/0.14001	**0.97998**/0.067128	0.96905/0.12786	0.97437/0.078144	0.9748/0.078183	0.78828/0.26366
4.5	0.96467/0.13438	0.96098/0.15166	0.9586/0.1414	**0.97659**/0.07018	0.97173/0.12469	0.96104/0.14354	0.9739/0.081547	0.76852/0.30043
5	0.96421/0.13409	0.95952/0.13361	0.9562/0.14408	**0.97712**/0.071431	0.96671/0.12969	0.96286/0.13376	0.97372/0.081887	0.78648/0.31082
Full Noise								
0.5	0.69136/0.38629	0.6794/0.40166	0.84603/0.31875	0.93636/0.16885	0.85152/0.31491	0.82473/0.34667	**0.93958**/0.16991	0.40663/0.35616
1	0.8063/0.33727	0.77118/0.36116	0.92839/0.20417	**0.96172**/0.088804	0.92952/0.20255	0.91416/0.22978	0.94817/0.14216	0.54788/0.36227
1.5	0.90595/0.24054	0.86855/0.28608	0.91152/0.23385	0.95842/0.096059	0.91902/0.22433	0.91271/0.24852	**0.9623**/0.1018	0.63036/0.31624
2	0.92239/0.22925	0.91427/0.22989	0.93627/0.20181	0.97079/0.076664	0.94004/0.19981	0.94978/0.16892	**0.97097**/0.078389	0.6036/0.33016
2.5	0.93629/0.20304	0.92795/0.20468	0.95552/0.14036	**0.97297**/0.06307	0.95993/0.13288	0.95622/0.13736	0.97049/0.081769	0.61713/0.35311
3	0.94707/0.17329	0.92711/0.20545	0.95839/0.13468	0.96853/0.084315	0.95878/0.13433	0.96163/0.13012	**0.97248**/0.073052	0.67621/0.30009
3.5	0.94722/0.17489	0.93333/0.20394	0.95125/0.14701	0.97084/0.072825	0.9609/0.1289	0.9525/0.14342	**0.97107**/0.081445	0.68242/0.32538
4	0.95943/0.1376	0.95312/0.13887	0.95874/0.13301	**0.96945**/0.077951	0.95947/0.1329	0.95769/0.13511	0.96865/0.081936	0.69334/0.30925
4.5	0.96206/0.13447	0.94222/0.18949	0.95778/0.13925	**0.97123**/0.071788	0.95955/0.13134	0.94832/0.14269	0.96261/0.092672	0.71631/0.27377
5	0.95839/0.1369	0.9473/0.1696	0.96225/0.12799	0.97228/0.079897	**0.97575**/0.074906	0.95835/0.13756	0.97074/0.086635	0.71862/0.29975