

# Nonnegative Matrix Analysis for Data Clustering and Compression

THESIS SUBMITTED  
IN ACCORDANCE WITH THE REQUIREMENTS OF  
THE UNIVERSITY OF LIVERPOOL  
FOR THE DEGREE OF  
DOCTOR IN PHILOSOPHY

BY

Liyun Gong

Department of Electrical Engineering and Electronics  
The University of Liverpool  
Liverpool, United Kingdom

February 9, 2015

# Abstract

Nonnegative matrix factorization (NMF) has becoming an increasingly popular data processing tool these years, widely used by various communities including computer vision, text mining and bioinformatics. It is able to approximate each data sample in a data collection by a linear combination of a set of nonnegative basis vectors weighted by nonnegative weights. This often enables meaningful interpretation of the data, motivates useful insights and facilitates tasks such as data compression, clustering and classification. These subsequently lead to various active roles of NMF in data analysis, e.g., dimensionality reduction tool [11, 75], clustering tool [94, 82, 13, 39], feature engine [40], source separation tool [38], etc.

Different methods based on NMF are proposed in this thesis: The modification of k-means clustering is chosen as one of the initialisation methods for NMF. Experimental results demonstrate the excellence of this method with improved compression performance. Independent principal component analysis (IPCA) which combines the advantage of both principal component analysis (PCA) and independent component analysis (ICA) has been chosen as the significant initialisation method for NMF with improved clustering accuracy. We have proposed the new evolutionary optimization strategy for NMF driven by three proposed update schemes in the solution space, saying NMF rule (or original movement), firefly rule (or beta movement) and survival of the fittest rule (or best movement). This proposed update strategy facilitates both the clustering and compression problems by using the different system objective functions that make use of the clustering and compression quality measurements. A hybrid initialisation approach is used by including the state-of-the-art NMF initialization methods as seed knowledge to increase the rate of convergence. There is no limitation for the number and the type of the initialization methods used for the proposed optimisation approach. Numerous computer experiments using the benchmark datasets verify the theoretical results, make comparisons among the techniques in measures of clustering/compression accuracy. Experimental results demonstrate the excellence of these methods with improved clustering/compression performance.

In the application of EEG dataset, we employed several standard algorithms to provide clustering on preprocessed EEG data. We also explored ensemble clustering to obtain some tight clusters. We can make some statements based on the results we have got:

firstly, normalization is necessary for this EEG brain dataset to obtain reasonable clustering; secondly, k-means, k-medoids and HC-Ward provide relatively better clustering results; thirdly, ensemble clustering enables us to tune the tightness of the clusters so that the research can be focused.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>xi</b>
<b>Acknowledgement</b>	<b>xii</b>
<b>Nomenclature</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Objectives . . . . .	1
1.2 Original Contribution . . . . .	2
1.3 Publications . . . . .	3
<b>2 Clustering Analysis</b>	<b>5</b>
2.1 Machine Learning . . . . .	5
2.2 Clustering Algorithms . . . . .	7
2.2.1 Problem setting . . . . .	8
2.2.2 Popular Clustering Algorithms . . . . .	10
2.2.3 Fuzzy clustering . . . . .	13
2.2.4 Kernel-based clustering . . . . .	14
2.2.5 Self Organizing Clustering . . . . .	15
2.2.6 Self Splitting and Merging Clustering . . . . .	17
2.2.7 Ensemble Consensus Clustering . . . . .	18
2.3 Clustering validation . . . . .	20
2.3.1 Internal Evaluation . . . . .	20
2.3.2 External Evaluation . . . . .	22
2.4 Summary . . . . .	24
<b>3 Nonnegative Matrix Factorization</b>	<b>25</b>
3.1 Introduction . . . . .	25
3.2 NMF optimization strategies . . . . .	27

3.2.1	Multiplicative update rule . . . . .	27
3.2.2	Other update rules . . . . .	28
3.3	NMF Initialization . . . . .	29
3.3.1	Random initialization . . . . .	29
3.3.2	Cluster-based Initialization . . . . .	30
3.3.3	Dimensionality reduction-based initialization . . . . .	34
3.3.3.1	Standard Dimensionality Reduction Methods . . . . .	34
3.3.3.2	PCA and ICA based Initialization . . . . .	38
3.3.4	Why initialization . . . . .	40
3.4	Summary . . . . .	41
<b>4</b>	<b>Proposed NMF Initialization Strategy</b>	<b>42</b>
4.1	Introduction . . . . .	42
4.2	Clustering Based Initialization . . . . .	43
4.2.1	Method . . . . .	44
4.2.2	Results . . . . .	44
4.2.3	Conclusion . . . . .	53
4.3	Dimensionality Reduction Based Initialization . . . . .	54
4.3.1	IPCA Based NMF . . . . .	54
4.3.2	Results . . . . .	55
4.3.3	Conclusion . . . . .	66
<b>5</b>	<b>Proposed NMF Updating Strategy for Data Clustering</b>	<b>68</b>
5.1	Introduction . . . . .	68
5.2	Proposed Method . . . . .	69
5.2.1	Seed Matrix Generation . . . . .	69
5.2.2	Evolving Strategy . . . . .	70
5.2.2.1	Multiplicative rule . . . . .	70
5.2.2.2	Firefly Rule . . . . .	70
5.2.2.3	Survival of the Fittest Rule . . . . .	73
5.2.3	Score Function . . . . .	74
5.3	Experimental results and analysis . . . . .	74
5.3.1	Experimental Setup . . . . .	75
5.3.2	Results and Analysis . . . . .	76
5.4	Conclusion . . . . .	87
<b>6</b>	<b>Proposed NMF Updating Strategy for Data Compression</b>	<b>89</b>
6.1	Introduction . . . . .	89
6.2	Proposed evolutionary optimization strategy on data compression . . . . .	91
6.2.1	Original movement for $\mathbf{W}$ . . . . .	93

6.2.2	Beta movement for $\mathbf{W}$ . . . . .	94
6.2.3	Best movement for $\mathbf{W}$ . . . . .	94
6.2.4	Objective function . . . . .	96
6.3	Experimental results and analysis . . . . .	97
6.3.1	Datasets and preprocessing . . . . .	97
6.3.2	Experiment process . . . . .	97
6.3.3	Yale image dataset . . . . .	98
6.3.4	ORL image dataset . . . . .	105
6.4	Conclusions . . . . .	112
<b>7</b>	<b>Application of EEG Dataset</b>	<b>117</b>
7.1	Introduction . . . . .	117
7.2	Clustering methods . . . . .	117
7.3	Experimental Results . . . . .	118
7.3.1	EEG data structure . . . . .	118
7.3.2	Results . . . . .	119
7.4	Conclusions . . . . .	133
<b>8</b>	<b>Summary</b>	<b>134</b>
8.1	Summary and conclusions . . . . .	134
8.2	Future Works . . . . .	135
	<b>Bibliography</b>	<b>146</b>

# List of Figures

2.1	The single linkage (nearest neighbour) distance), from [30] . . . . .	12
2.2	The complete linkage distance, from [30] . . . . .	12
2.3	The average linkage distance, from [30] . . . . .	13
3.1	Description of NMF and SVD basis vectors on face dataset, from [23] . . . . .	26
3.2	Error using different initializations, from [122] . . . . .	32
3.3	Comparison of initialisation methods in terms of sparsity and orthogonality. (a) Comparison of initialisation methods in term of orthogonality and (b) Comparison of initialisation methods in term of sparsity . . . . .	33
3.4	Comparison of initialisation methods in terms of error . . . . .	33
3.5	Prostate cancer study: sample representation using the first two or three components from PCA, ICA and IPCA, from [118] . . . . .	38
4.1	The average $D$ log values of each of the five initialization methods (Dataset 1) . . . . .	47
4.2	The average $D$ log values of each of the five initialization methods (Dataset 2) . . . . .	48
4.3	The average $D$ log values of each of the five initialization methods (Dataset 3) . . . . .	48
4.4	The average $D$ log values of each of the five initialization methods (Dataset 4) . . . . .	49
4.5	The average $D$ log values of each of the five initialization methods (Dataset 5) . . . . .	49
4.6	The average $D$ log values of each of the five initialization methods (Dataset 6) . . . . .	50
4.7	The average $D$ log values of each of the five initialization methods (Dataset 7) . . . . .	50
4.8	The average $D$ log values of each of the five initialization methods (Dataset 8) . . . . .	51
4.9	The average $D$ values of each of the five initialization methods with the increasing dimensionality number on Dataset 1-4 . . . . .	52

4.10	The average $D$ values of each of the five initialization methods with the increasing dimensionality number on Dataset 5-8 . . . . .	53
4.11	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for balance dataset . . . . .	58
4.12	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for cancer-int dataset . . . . .	59
4.13	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for credit dataset . . . . .	60
4.14	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for dermatology dataset . . . . .	61
4.15	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for diabetes dataset . . . . .	62
4.16	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for iris dataset . . . . .	63
4.17	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for thyroid dataset . . . . .	64
4.18	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for wdbc dataset . . . . .	65
4.19	The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for wine dataset . . . . .	66
5.1	Data flow of the proposed ENMF. The circle, triangle and rectangle symbols represent candidates derived during the generation of the $S_t^{(M)}$ , $S_t^{(F)}$ and $S_t^{(S)}$ subsets, respectively. . . . .	71
5.2	The accuracies of the corresponding testing dataset for balance (ENMF-MIX) . . . . .	79
5.3	The accuracies of the corresponding testing dataset for breast tissue (ENMF-MIX) . . . . .	80
5.4	The accuracies of the corresponding testing dataset for cancer (ENMF-MIX) . . . . .	81
5.5	The accuracies of the corresponding testing dataset for cancerint (ENMF-MIX) . . . . .	82
5.6	The accuracies of the corresponding testing dataset for dermatology (ENMF-MIX) . . . . .	83
5.7	The accuracies of the corresponding testing dataset for glass (ENMF-MIX) . . . . .	84
5.8	The accuracies of the corresponding testing dataset for haberman (ENMF-MIX) . . . . .	85
5.9	The accuracies of the corresponding testing dataset for iris (ENMF-MIX) . . . . .	86
5.10	The accuracies of the corresponding testing dataset for thyroid (ENMF-MIX) . . . . .	87



6.1	The general process of the proposed evolutionary optimization strategy	92
6.2	The flow chart of the proposed evolutionary optimization strategy . . .	93
6.3	The structure of the experimental process . . . . .	98
6.4	The summary results for Yale face dataset under the different data compression methods with the sparsity measure . . . . .	100
6.5	The summary results for Yale face dataset under the different data compression methods with the orthogonality measure . . . . .	101
6.6	The summary results for Yale face dataset under the different data compression methods with both the sparsity and orthogonality measures . .	102
6.7	The basis images obtained from different methods for Yale with the sparsity measure. (1) 1st row represents the basis images obtained from random initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (2) 2nd row represents the basis images obtained from random acol initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (3) 3rd row represents the basis images obtained from k-means based initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (4) 4th row represents the basis images obtained from FCM-based initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (5) 5th row represents the basis images obtained from the proposed evolutionary optimization strategy at the increasing iterations (iter=1, 50, 100, 200, 500) . . . . .	103
6.8	The basis images obtained from different methods for Yale with the orthogonality measure which is similar with figure6.7. . . . .	104
6.9	The basis images obtained from different methods for Yale with both the sparsity and orthogonality measures which is similar with figure6.7. . .	105
6.10	The summary results for ORL face dataset under the different data compression methods with the sparsity measure . . . . .	107
6.11	The summary results for ORL face dataset under the different data compression methods with the orthogonality measure . . . . .	108
6.12	The summary results for ORL face dataset under the different data compression methods with both the sparsity and orthogonality measures . .	109

6.13	The basis images obtained from different methods for ORL with the sparsity measure. (1) 1st row represents the basis images obtained from random initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (2) 2nd row represents the basis images obtained from random acol initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (3) 3rd row represents the basis images obtained from k-means based initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (4) 4th row represents the basis images obtained from FCM-based initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (5) 5th row represents the basis images obtained from the proposed evolutionary optimization strategy at the increasing iterations (iter=1, 100, 200, 500, 1000) . . . . .	110
6.14	The basis images obtained from different methods for ORL with the orthogonality measure which is similar with figure6.13. . . . .	111
6.15	The basis images obtained from different methods for ORL with both the sparsity and orthogonality measures which is similar with figure6.13.	112
7.1	The independent component matrix . . . . .	119
7.2	Histogram of k-means clustering result with $K = 64$ from the unnormalized data . . . . .	120
7.3	The largest cluster (No 33 cluster) in the k-means clustering result with $K = 64$ from the unnormalized data . . . . .	121
7.4	One of single-member clusters, No 31 cluster, in the k-means clustering result with $K = 64$ from the unnormalized data . . . . .	122
7.5	Histogram of k-means clustering result with $K = 64$ from the normalized data . . . . .	123
7.6	The profiles of the members in No.40 cluster in normalized data . . . . .	124
7.7	The profiles of the members in No.40 cluster in unnormalized data . . . . .	125
7.8	The profiles of the members in No.18 cluster in normalized data . . . . .	126
7.9	The profiles of the members in No.18 cluster in unnormalized data . . . . .	127
7.10	The profiles of the members in No.19 cluster in normalized data . . . . .	128
7.11	The profiles of the members in No.19 cluster in unnormalized data . . . . .	129
7.12	The list of clustering results we have got . . . . .	130
7.13	The mean and standard deviation of the numbers of memberships . . . . .	130
7.14	Number of clusters which contains ICs from more than seven ( $> 7$ ) subjects	131
7.15	The clusters containing more than seven subjects in the ensemble clustering results: $K = 64$ . . . . .	132
7.16	The clusters containing more than seven subjects in the ensemble clustering results: $K = 24$ . . . . .	133

# List of Tables

2.1	Kaufman Approach (KA) initialization method . . . . .	11
2.2	The procedure of FCM . . . . .	14
2.3	The procedure of kernel k-means and kernel FCM . . . . .	15
2.4	The procedure of SOON-2 . . . . .	17
3.1	The comparison between random and random acol intialization . . . . .	41
4.1	The parameters for the preprocessing . . . . .	45
4.2	The properties of the final datasets . . . . .	45
4.3	The range of the standard deviation for $\log_2(D)$ . . . . .	46
4.4	The properties of the datasets . . . . .	55
4.5	The mean and standard deviation of inital RAND values of different intialization methods (iters=1, runs=20) . . . . .	56
4.6	The mean and standard deviation of final RAND values of different in- tialization methods (iters=500, runs=20) . . . . .	56
5.1	Details of the datasets used. . . . .	75
5.2	Performance comparison for different datasets in RAND, which is re- ported in percentage (%). The best performance is highlighted in bold, and the second best is underlined. . . . .	75
5.3	Performance improvement of ENFM over NMF under different initial- ization approaches. . . . .	76
6.1	The procedure for finding the best solution $\mathbf{W}^{t^*}$ . . . . .	95
6.2	The summary results for Yale face dataset under the different data com- pression methods with the five measurements saying sparsity, orthog- onality, objective value, error and RAND index at the final iteration (iter=500) . . . . .	114
6.3	Statistical significance test (t-test) for the average ENMF and NMF after 5 runs shown in table 6.2 . . . . .	115
6.4	Statistical significance test (t-test) for the average ENMF and NMF after 5 runs shown in table 6.5 . . . . .	115

6.5	The summary results for ORL face dataset under the different data compression methods with the five measurements saying sparsity, orthogonality, objective value, error and RAND index at the final iteration (iter=500) . . . . .	116
-----	--	-----

# Acknowledgement

I would like to express my deep gratitude to Dr. Tingting Mu, Prof.Nandi Asoke and Dr. Al-Nuaimy Waleed, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work.

# Nomenclature

- KA** Kaufman Approach
- FCM** fuzzy c-means
- FSOM** fuzzy self-organizing map
- FART** fuzzy adaptive resonance theory
- FSVM** fuzzy support vector machine
- SVM** support vector machines
- SOM** self-organizing map
- SOON** self-organizing oscillator networks
- OPTOC** one-prototype-take-one-cluster
- SSMCL** self-splitting-merging competitive learning
- CSM** cohesion-based self-merging
- APV** asymptotic property vector
- PVI** parametric validity index
- MST** mining spanning tree
- NMF** Nonnegative Matrix factorization
- SVD** singular value decomposition
- HC** hierarchical clustering
- RI** random initialization
- RAI** random acol initialization
- RCI** random c initialization
- PCA** principal component analysis

**ICA** independent component analysis

**IPCA** independent principal component analysis

**FA** firefly algorithm

# Chapter 1

## Introduction

### 1.1 Motivation and Objectives

Machine learning is a subfield of computer science and artificial intelligences that deals with the design and development of systems. Over the last few decades, there have been many significant advances in machine learning. More and more applications of machine learning are found in many different fields like biomedical engineering and communications. Machine learning is very important not only because that the achievement of learning in machines might help us understand how animal and humans learn, but also the following engineering reasons [76]. We might be able to specify input/output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and, thus, suitably constrain their input/output function to approximate the relationship implicit in the examples. Also, machine learning can be used to reach on-the-job improvement of existing machine designs, to capture more knowledge than what humans would want to write down, to adapt to a changing environment to reduce the need for constant redesign, and to track as much new knowledge as possible.

Clustering is one of the most useful tools for the large data analysis and is my first research topic. It is known as unsupervised learning which is to group individual objects or samples in a population within which the objects are more similar to each other than those in other clusters. It has been used for decades in many fields, such as image processing, data mining, artificial intelligence [113] and the microarray gene expression data analysis in genomic research [48].

Compression is another useful tool for machine learning. It involves encoding information using fewer bits than the original representation [77]. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by identifying unnecessary information and removing it [46]. The process of reducing the size of a data file is popularly referred to as data compression and is



also one of my research topics. Data compression is also widely used in backup utilities, spreadsheet applications, and database management systems.

The objectives of this thesis are developing machine learning algorithms for clustering, classification and compression, emphasis on nonnegative matrix factorization (NMF). Nonnegative matrix factorization (NMF) has becoming an increasingly popular data processing tool these years, widely used by various communities including computer vision, text mining and bioinformatics. It is able to approximate each data sample in a data collection by a linear combination of a set of nonnegative basis vectors weighted by nonnegative weights. This often enables meaningful interpretation of the data, motivates useful insights and facilitates tasks such as data compression, clustering and classification. These subsequently lead to various active roles of NMF in data analysis, e.g., dimensionality reduction tool [11, 75], clustering tool [94, 82, 13, 39], feature engine [40], source separation tool [38], etc. In this research work, the NMF algorithm is explored, emphasis being laid on the topic of the initialization methods as well as the optimization rule for NMF, to solve some data analysis problems. We propose two initialization methods for NMF based on the clustering algorithm and dimensionality reduction algorithm. We also propose two NMF updating strategies, which take advantage of the hybrid of different NMF initialization setups and evolves along different directions to produce NMF approximations that suits better the accuracy purposes (i.e. data clustering/classification and compression). Effectiveness of the proposed methods is demonstrated thoroughly through benchmark testing and comparison with existing approaches.

## 1.2 Original Contribution

A summary of the the main original contributions of this work are shown below on a chapter-by-chapter basis.

### Chapter 2

We review five clustering families representing five clustering concepts including fuzzy clustering, kernel-based clustering, self-organizing clustering, self-splitting and merging clustering and ensemble clustering.

### Chapter 4

We propose two initialization methods for NMF based on the clustering algorithm and dimensionality reduction algorithm. The modification of k-means clustering and independent principal component analysis (IPCA) are chosen as the initialisation methods for NMF.

## **Chapter 5**

We propose the new evolutionary optimization strategy for NMF driven by three proposed update schemes in the solution space, saying NMF rule, firefly rule and survival of the fittest rule. This proposed update strategy facilitates the clustering problem by using the system objective functions that make use of the clustering quality measurements.

## **Chapter 6**

We propose the new evolutionary optimization strategy for NMF by modifying the proposed optimization strategy in chapter 5 to solve the image datasets compression problem.

## **Chapter 7**

we employ several standard algorithms to provide clustering on the application of pre-processed EEG data. We also explore ensemble clustering to obtain some tight clusters.

## **1.3 Publications**

Papers arisen from the this PhD work are listed as follows:

### **Journal Papers**

1. Liyun Gong, Tingting Mu, Meng Wang, Hengchang Liu and John Y. Goulermas, A robust nonnegative matrix factorization strategy with adaptive quality control of data clusters, *Pattern Recognition*, 2014. (IF=2.632, under review)
2. FY Cong, V Alluri, AK Nandi, P Toiviainen, Rui Fa, Basel Abu-Jamous, Liyun Gong, BGW Craenen, H Poikonen, M Huotilainen, T Ristaniemi, Linking Brain Responses to Naturalistic Music through Analysis of Ongoing EEG and Stimulus Features, *IEEE Transactions on Multimedia*, 15(5):1060-1069, 2013. (IF=1.776)

### **Conference Papers**

1. Liyun Gong, T. Mu and Al-Nuaimy Waleed, Evolutionary nonnegative matrix factorization for data compression, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2015. (to be submitted)
2. Liyun Gong and Asoke K. Nandi, Clustering by Non-negative Matrix Factorization with Independent Principal Component Initialization. *European Signal Processing Conference, EUSIPCO*, 2013.(acceptance rate=62.5%)

3. Liyun Gong and Asoke K. Nandi, An Enhanced Initialization Method for Non-negative Matrix Factorization, IEEE International Workshop on Machine Learning for Signal Processing, MLSP, 2013. (acceptance rate=51%)
4. Rui Fa, Asoke K Nandi and Liyun Gong, Clustering analysis for gene expression data: a methodological review, 5th international symposium on communication, control, and signal processing, ISCCSP, 2012. (acceptance rate=48%)

## Chapter 2

# Clustering Analysis

This chapter describes the basis knowledge of clustering analysis and also reviews some common clustering algorithms. Firstly section 2.1 provides a brief introduction to machine learning, including supervised learning, unsupervised learning and semi-supervised learning. Section 2.2 provides the comprehensive review of some popular clustering algorithms in addition with the five clustering families representing five clustering concepts. Since clustering is one of the most widely used unsupervised learning technique, the task of assessing the results of clustering algorithms can be as important as the clustering algorithms themselves. So several clustering validations are reviewed including internal and external evaluations as well in section 2.3. The reason for this chapter is that some methods described here have been selected to solve the brain application in chapter 7 and also some of them were treated as the initialisation methods for nonnegative matrix factorisation in chapter 5 and 6. Work of this chapter is published in:

Rui Fa, Asoke K Nandi and Liyun Gong, Clustering analysis for gene expression data: a methodological review, 5th international symposium on communication, control, and signal processing, ISCCSP, 2012. (acceptance rate=48%)

### 2.1 Machine Learning

Machine learning is a subfield of computer science and artificial intelligences that deals with the design and development of systems. It has been defined formally by Mitchel [71] as "A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $M$ , if its performance at tasks in  $T$ , as measured by  $M$ , improves with experience  $E$ ." Over the past 50 years, the study of machine learning has grown from the efforts of a handful of computer engineers exploring whether computers could learn to play games, and a field of statistics that largely ignored computational considerations, to a broad discipline that has produced fundamental statistical-computational theories of learning processes; has designed learning algorithms that are routinely used in commercial systems from speech recognition to

computer vision; and has spun off an industry in data mining to discover hidden regularities in the growing volume of online data [72]. A number of choices are involved in designing a machine learning approach, including choosing the type of training experience, the target function to be learned, a representation for this target function, and an algorithm for learning the target function from training samples [71]. Machine learning is inherently a multidisciplinary field, which draws on results from artificial intelligence, probability and statistics, optimization theory, computational complexity theory, control theory, information theory, philosophy, and other fields.

Machine learning has been used in many applications including natural language processing [22, 68, 74], handwriting recognition [60, 79, 80, 84], face and fingerprint recognition [47, 79, 80, 116], bioinformatics and cheminformatics [8, 42, 110], object recognition in computer vision [102], image compression [114].

### **Types of Algorithms**

Machine learning algorithms are organised into several forms including supervised learning, unsupervised learning and semi-supervised learning.

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. Each sample consists of a pair of an input object (typically feature vector) and a desired output (target). The output of the function can be used for calculating new examples (regression), or can determine the class labels for unseen input objects (classification). In order to solve a given problem of supervised learning, one has to consider six issues.

1. Determine the type of training samples.
2. Gathering a training set that contains information of problem. Thus, a set of input objects and the corresponding outputs are gathered, either from human experts or from measurements.
3. Determine the input feature representation of the learned function (feature extraction). The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should be large enough to predict the output accurately.
4. Determine the structure of the learned function and corresponding learning algorithm.
5. Complete the design. The engineer then runs the learning algorithm on the

gathered training set. Parameters of the learning algorithm may be adjusted by optimizing performance on a subset of the training set (called a validation set) or via cross-validation. After parameter adjustment and learning, the performance of the algorithm may be measured on a test set that is separate from the training set.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

Unsupervised learning is the other form of machine learning which tries to find hidden structure in unlabelled data. It is distinguished from supervised learning by the fact that there is no a priori output. In unsupervised learning, a data set of input objects is gathered, and treated as a set of random variables. A joint density model is then built for the data set. Unsupervised learning can be used in conjunction with Bayesian inference to produce conditional probabilities for any of the random variables given the others. A holy grail of unsupervised learning is the creation of a factorial code of the data, which may make the later supervised learning method work better when the raw input data is first translated into a factorial code. Unsupervised learning is also useful for data compression. Another form of unsupervised learning is clustering which is introduced in details later.

Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training—typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabelled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent. The cost associated with the labelling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabelled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value.

## 2.2 Clustering Algorithms

Clustering is one of the most useful tools for the large data analysis and is the one of my research topics. It is known as unsupervised learning and has been used for decades in many fields, such as image processing, data mining, artificial intelligence [113] and the microarray gene expression data analysis in genomic research [48]. The goal of the clustering analysis is to group individual objects or samples in a population

within which the objects are more similar to each other than those in other clusters. Generally speaking, to study or design a clustering analysis for an application, one has to consider three issues:

1. the measurements of the dissimilarity (or similarity)
2. the clustering algorithms
3. the clustering validations

There has been a rich literature on clustering analysis over the past decades and all these three issues have been comprehensively discussed in [48] and [113]. However, as it has been a few years since those two comprehensive review papers [48, 113] were published, many new and effective algorithms have been proposed but were not reviewed. Both the clustering algorithms and validations have grown beyond the horizon of [48, 113]. The following sections can be viewed as a complementary counterpart to make the literature review in this field some-how up-to-date.

In section 2.2.2, we review some popular clustering algorithms, saying k-means, k-medoids and hierarchical clustering. Besides some popular clustering algorithms, we will also discuss five different families of clustering algorithms including fuzzy clustering, kernel-based clustering, self-organizing clustering, self-splitting and merging clustering and ensemble clustering in section 2.2.3-2.2.7.

### 2.2.1 Problem setting

There were seven similarity and dissimilarity measures listed in [113], namely, *Minkowski distance*, *Euclidean distance*, *City-block distance*, *Mahalanobis distance*, *Pearson correlation*, *Point symmetry distance*, *Cosine similarity*, which have been widely used in various applications. In [48], *Euclidean distance* and *Pearson correlation* were claimed to be effective similarity measures for gene expression data. *Pearson correlation* measures the similarity between two genes, and provides a very informative visualisation of the clustering results. Based on a sample of paired genes  $(X, Y)$ , the pearson correlation is:

$$PC = \frac{1}{n-1} \sum_{i=1}^n \left( \frac{X_i - \bar{X}}{S_X} \right) \left( \frac{Y_i - \bar{Y}}{S_Y} \right) \quad (2.1)$$

where

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i; \quad (2.2)$$

$$S_X = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2} \quad (2.3)$$

are the mean and the standard deviation for  $X$  and  $n$  is the number of dimensions for  $X$ . (The terms for  $Y$  are similar.)

*Minkowski distance* and *City-block distance* (also called *Manhattan distance*) are the

metrics on Euclidean space which can be considered as the generalization of the *Euclidean distance*. The *Minkowski distance* of order  $p$  between two points  $X = (x_1, x_2, \dots, x_n)$  and  $Y = (y_1, y_2, \dots, y_n)$  is defined as:

$$MinD = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (2.4)$$

*Minkowski distance* is typically used with  $p$  being 1 or 2. The latter is the Euclidean distance, while the former is known as *City-block distance*.

*Mahalanobis distance* is equated to the *euclidean distance* in a transformed whitened space. Given an sample  $x = (x_1, x_2, x_3, \dots, x_N)^T$  from a group of samples with mean  $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_N)$ , this distance is defines as:

$$MahD(x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)} \quad (2.5)$$

where  $S$  is the covariance matrix of this group of samples. If  $S$  is the identity matrix, *Mahalanobis distance* reduces to *euclidean distance*.

*Point symmetry distance* is the distance which incorporates both the Euclidean distance as well as a measure of symmetry. Given  $N$  samples  $x_i, i = 1, \dots, N$  and a reference sample  $c$ , the distance between a sample  $x_j$  and the reference sample  $c$  is defined as:

$$PD(x_j, c) = \min_{j=1, \dots, N; i \neq j} \frac{\|(x_j - c) + (x_i - c)\|}{\|x_j - c\| + \|x_i - c\|} \quad (2.6)$$

where the denominator term is used to normalize the point symmetry distance so as to make the point symmetry distance insensible to the Euclidean distances  $\|x_j - c\|$  and  $\|x_i - c\|$ .

*Cosine similarity* is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. Given two samples/vectors  $X$  and  $Y$ , the cosine similarity,  $\cos(\theta)$ , is calculated using the dot product and magnitudes of  $X$  and  $Y$  as

$$CosS = \cos(\theta) = \frac{X \cdot Y}{\|X\| \|Y\|} \quad (2.7)$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity.

Furthermore, two additional measures, namely *Jackknife correlation* and *Spearman's rank-order correlation*, were discussed to cope with the situations of outliers and non-Gaussian distributions, respectively. In the following sections of this chapter, we will not discuss the dissimilarity and similarity measures but use the operators  $D(\cdot)$  for dissimilarity and  $S(\cdot)$  for similarity instead of a specific measure when we study clustering algorithms. Readers who are interested in more details are advised to refer to [113, 48] and the references therein.



Here we suppose that we are going to partition the dataset  $X = \{x_n | 1 \leq n \leq N\}$ , where  $x_n \in \mathbb{R}^{M \times 1}$  denotes the  $n$ -th object,  $M$  is the number of samples (features or dimensions) and  $N$  is the number of objects. Consider that there are  $K$  clusters in a given dataset and each clustering algorithm provides a partition matrix  $U^{K \times N}$ , where the entry  $u_{k,n} \in [0, 1]$  represents the membership coefficient of  $n$ -th object in the  $k$ -th cluster.

## 2.2.2 Popular Clustering Algorithms

### K-means Clustering

The k-means clustering algorithm is one of the simplest and most common partitioning methods.[64, 66]. For the traditional k-means, it starts with the  $k$  cluster centers chosen randomly, where  $k$  is the cluster number. Using distance methods such as Euclidean distance to measure the similarity between data objects and the cluster centers. Thus, it leads to the following objective function:

$$E = \sum_{j=1}^K \sum_{x_i \in C_j} \|x_i - u_j\|^2 \quad (2.8)$$

where  $C_j$  denotes the  $j$ -th cluster,  $u_j$  is the center of the cluster  $j$  which is the mean of objects in  $C_j$  and  $x_i$  is the observations to be clustered. Each object is then assigned to one of the cluster groups with the closest center. Then the cluster centers are redefined by finding the mean vector of all objects belonging to each cluster group and the objects are reassigned according to their distance to these new cluster centers. This iterative process repeats until there are no changes in the assignment of objects to cluster groups. The algorithm is often presented as assigning objects to the nearest cluster by distance. The standard algorithm aims at minimizing the Euclidean objective, and thus assigns by "least sum of squares". Using a different distance function other than the squared Euclidean distance may stop the algorithm from converging. Various modifications of k-means such as spherical k-means [89] and k-medoids [52] have been proposed to allow using other distance measures. For example, k-medoids chooses objects as centers and works with an arbitrary matrix of distances between objects instead of  $L_2$ . This method was proposed in [52] for the work with  $L_1$  norm and other distances.

Since different starting points can result in the different cluster results, it may be advisable to run the algorithm several times and select the best solution among them. Also in literature, some initialization algorithms have been proposed to the traditional k-means clustering to avoid the influence of the randomness [83]. Here we review the Kaufman Approach (KA) proposed by Kaufman and Rousseeuw [53] which is the famous initialization method for k-means. The details are shown in table 2.1.

Table 2.1: Kaufman Approach (KA) initialization method

<p>STEP1: Select the first center so that it has the minimum distance to the other objects.</p> <p>STEP2: For every nonselected object <math>w_j</math>: Calculate <math>C_{ji} = \max(D_j - d_{ji}, 0)</math>, where <math>d_{ji}</math> is the absolute distance between <math>w_i</math> and <math>w_j</math>. <math>w_i</math> is the second randomly selected center.</p> <p><math>D_j</math> is the minimum distance between the pre-selected centers and <math>w_j</math>.</p> <p>STEP3: Calculate <math>\sum_j C_{ji}</math> for the current <math>w_i</math>.</p> <p>STEP4: Repeat STEP2 and STEP3 and select the second center <math>w_i</math> which maximizes <math>\sum_j C_{ji}</math>.</p> <p>STEP5: Iterative process continues until <math>k</math> initial centers are selected.</p>
--

The main disadvantage of the k-means clustering algorithm is the predefined cluster number. It is difficult to set the number of cluster since most of the real datasets are unknown.

### K-medoids

The k-medoids is one of many partitioning algorithms, which is an extension of the k-means. The k-medoids is designed to handle the outliers efficiently. Instead of using the means, it chooses the medoids to represent the cluster centers. A medoid is the most centrally located object within a cluster, whose total distance to all other objects intra-cluster is the shortest. Similar to the k-means, the k-medoids keeps updating the medoids if cluster membership changes until the process converges.

### Hierarchical Clustering (HC)

In contrast to partition-based clustering, which directly divides the data set into disconnected parts, HC is path-based clustering algorithm, which generates a hierarchical series of nested and connected clusters. This nested cluster structure can be graphically represented in a tree by *dendrogram*. There are two approaches to implement the HC: one is called agglomerative which initially regards each data object as an individual cluster and merges the closest pair of clusters at each step, until all the groups are merged into one cluster; The other is called divisive, which starts with one cluster containing all the data objects and splits singleton clusters at each step. In this thesis, we employ agglomerative clustering. HC provides different clustering results when employing different linkage criteria. The linkage criteria determine the distance between clusters as a function of the pairwise distances between objects.

**Single Linkage** uses the distance between the nearest neighbours. For example, if we have two clusters  $A$  and  $B$ , the single linkage distance is defined as

$$d(A, B) = \min_{\alpha=1, \dots, n_A \beta=1, \dots, n_B} d(a_\alpha, a_\beta) \quad (2.9)$$

$n_A$  and  $n_B$  are the number of points in cluster  $A$  and  $B$  respectively.  $d(\cdot)$  is the distance

between two points.

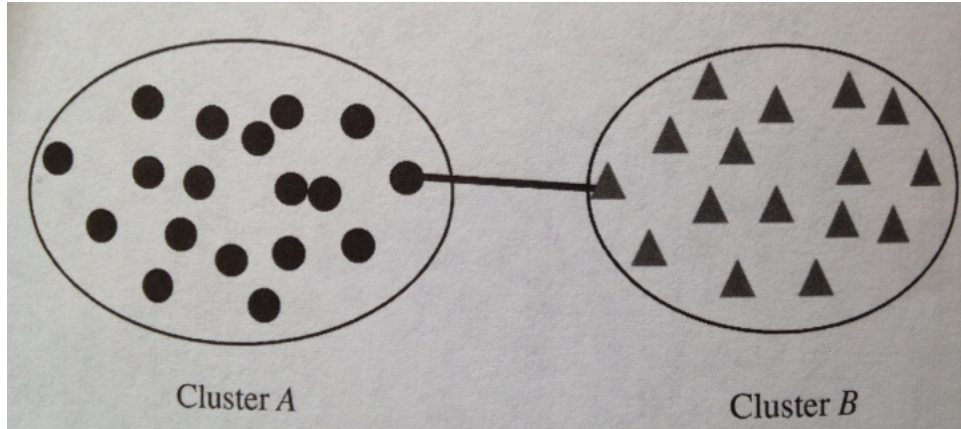


Figure 2.1: The single linkage (nearest neighbour) distance), from [30]

**Complete linkage** uses the maximum distance between objects.

$$d(A, B) = \max_{\alpha=1, \dots, n_A, \beta=1, \dots, n_B} d(a_\alpha, a_\beta) \quad (2.10)$$

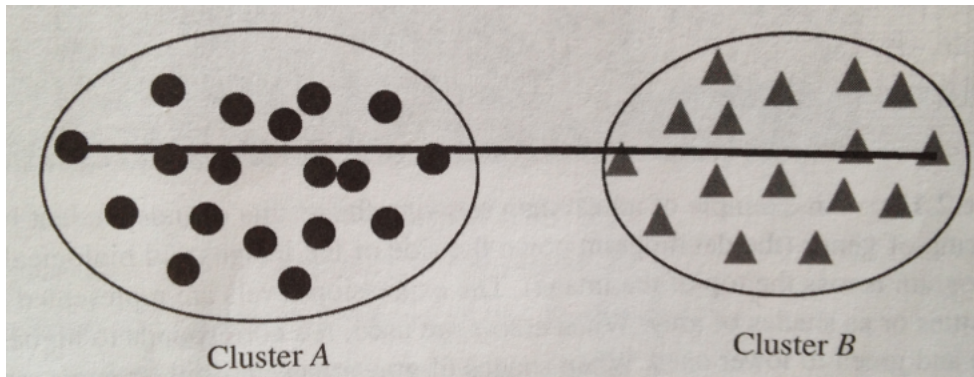


Figure 2.2: The complete linkage distance, from [30]

**Average linkage** uses average distance of objects.

$$d(A, B) = \frac{1}{n_A n_B} \sum_{\alpha=1}^{n_A} \sum_{\beta=1}^{n_B} d(a_\alpha, a_\beta) \quad (2.11)$$

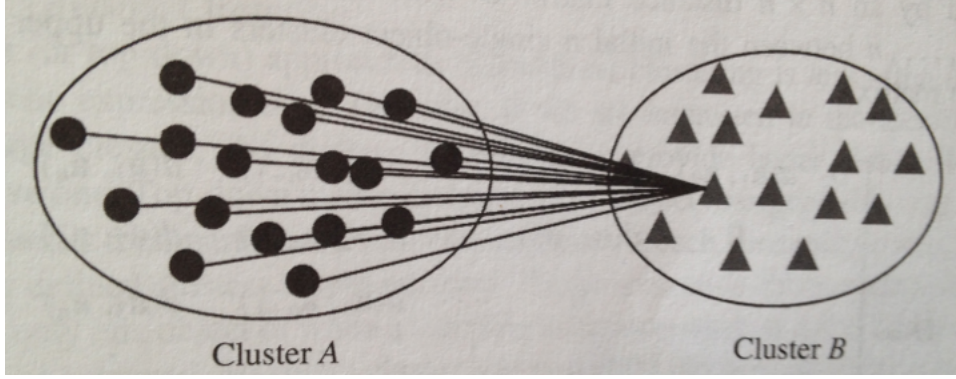


Figure 2.3: The average linkage distance, from [30]

**Ward Linkage:** In ward linkage, the distance between two clusters is the ANOVA sum of squares between the two clusters added up over all the objects.

There are other linkage such as centroid linkage which uses the variance of the merging clusters.

### 2.2.3 Fuzzy clustering

Fuzzy clustering is a concept that relaxes the restriction of crisp clustering, which assigns every object exactly in one clusters, by characterizing the membership of each sample point in all the clusters with a membership function which ranges between zero and one. Additionally, the sum of the memberships for each sample point must be unity [9]. The properties of the partition matrix is mathematically expressed by

1.  $u_{k,n} \in [0, 1], \quad 1 \leq n \leq N, 1 \leq k \leq K$
2.  $\sum_{k=1}^K u_{k,n} = 1, \quad 1 \leq n \leq N$
3.  $0 < \sum_{n=1}^N u_{k,n} < N, \quad 1 \leq k \leq K$

An example of fuzzy clustering is the fuzzy c-means (FCM) [10], which is fuzzy counterpart of the k-means. FCM aims to minimize the cost function, which is mathematically expressed by

$$\phi(U, X) = \sum_{k=1}^K \sum_{n=1}^N (u_{k,n})^m D(x_n, c_k) \quad (2.12)$$

where  $m \in [1, \infty)$  is the fuzzification parameter.  $D(x_n, c_k)$  is the dissimilarity measure between the  $n^{th}$  object,  $x_n$ , and the center of the  $k^{th}$  cluster,  $c_k$ . Similar to k-means, the cost function 2.12 can be minimised with an iterative procedure that updates the partition matrix and the centers of clusters alternately. Since many objects may participate in more than one function in the process, the fuzzy clustering has obvious advantage to identify some objects co-regulating with more than one cluster of objects. The procedure of FCM is summarized in Table 2.2. There are also many other fuzzy clustering approaches, for example, fuzzy self-organizing map (FSOM) [81], fuzzy adaptive resonance theory (FART) [101] and fuzzy support vector machine (FSVM) [67].

Table 2.2: The procedure of FCM

<p>STEP1: Initialize the centers of clusters <math>\{c_k k = 1, \dots, K\}</math> randomly or based on some prior knowledge if available;</p> <p>STEP2: Update the membership matrix <math>U</math> by <math>u_{k,n} = 1 / \left[ \sum_{k'=1}^K \left( \frac{D(x_n, c_{k'})}{D(x_n, c_k)} \right)^{2/(1-m)} \right]</math>, <math>k = 1, \dots, K</math> and <math>n = 1, \dots, N</math>.</p> <p>STEP3: Update the centroids of clusters <math>\{c_k k = 1, \dots, K\}</math> by <math>c_k(t) = \frac{\sum_{n=1}^N u_{k,n}^m x_n}{\sum_{n=1}^N u_{k,n}^m}</math> for <math>k = 1, \dots, K</math>.</p> <p>STEP4: Repeat <b>STEP2 and 3</b> until <math>\ C(t) - C(t-1)\  &lt; \epsilon</math>, where <math>\epsilon</math> is a small positive number.</p>
---

## 2.2.4 Kernel-based clustering

Kernel-based clustering, which shares similar idea with support vector machines (SVM), constructs a hyperplane to separate the patterns. These patterns are nonlinearly transformed from a set of nonlinearly separable patterns into a higher-dimensional feature space to be linear separable. At the core of the kernel-based clustering lies the difficulty of explicitly constructing the nonlinear mapping,  $\Phi(\cdot)$ , which is sometime infeasible; but now it can be overcome by a kernel trick. The kernel trick is a way of mapping patterns from a input space into a feature space without having to compute the mapping explicitly, in the hope that the patterns will gain meaningful linear structure in the feature space, mathematically expressed as

$$k(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) \quad (2.13)$$

where  $(\cdot)^T$  is the transpose operator. Thus, a straightforward way to transform the calculation of Euclidean distance in the feature space into the kernel version is to use the kernel trick as follows.

$$\begin{aligned} D_E^k(\Phi(x_i), \Phi(x_j)) &= \|\Phi(x_i) - \Phi(x_j)\|^2 \\ &= \|\Phi(x_i)\|^2 + \|\Phi(x_j)\|^2 - 2\Phi(x_i)^T \Phi(x_j) \\ &= k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j) \end{aligned} \quad (2.14)$$

Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. It shows the linear relationship between two sets of data. The kernel version of modified Pearson correlation is given by [86].

$$S_P^k(\Phi(x_i), \Phi(x_j)) = \frac{\Phi(x_i)^T \Phi(x_j)}{\sqrt{\Phi(x_i)^T \Phi(x_i)} \sqrt{\Phi(x_j)^T \Phi(x_j)}} = \frac{k(x_i, x_j)}{\sqrt{k(x_i, x_i)} \sqrt{k(x_j, x_j)}} \quad (2.15)$$

Kernel k-means (or kernel FCM) is kernel counterpart of the k-means (or FCM) [26], whose core part is to calculate the distance between the objects and the centroids of clusters in the feature space.

$$D_E^k(\Phi(x_i), c_k^\Phi) = \|\Phi(x_i) - \frac{1}{N_k} \sum_{l=1}^N u_{k,l}^m \Phi(x_l)\|^2 = k(x_i, x_i) - \frac{1}{N_k} \sum_{l=1}^N u_{k,l}^m k(x_i, x_l) + \frac{1}{N_k^2} \sum_{l=1}^N \sum_{n=1}^N u_{k,l}^m u_{k,n}^m k(x_l, x_n) \quad (2.16)$$

where  $N_k = \sum_{l=1}^N u_{k,l}^m$ . For crisp kernel k-means, the elements in partition matrix are either zero or one and  $m = 1$ ; for the kernel FCM, the partition matrix is the one described in Section 2.2.3 and  $m \in [1, \infty)$ . The procedure of kernel k-means and kernel FCM is summarised in Table 2.3. Other kernel-based algorithms including kernel hierarchical and kernel principle component analysis can be found in [86, 63]

Table 2.3: The procedure of kernel k-means and kernel FCM

<p><b>STEP1:</b> Initialize a K-partition in the feature space;</p> <p><b>STEP2:</b> Calculate <math>D_E^k(\Phi(x_n), c_k^\Phi)</math> for <math>n = 1, \dots, N</math> and <math>k = 1, \dots, K</math>.</p> <p><b>STEP3:</b> Update the membership matrix <math>U(t)</math> by</p> $u_{k,n} = \begin{cases} 1 & D_E^k(\Phi(x_n), c_k^\Phi) < D_E^k(\Phi(x_n), c_{k'}^\Phi) \\ 0 & \text{otherwise} \end{cases} \quad (2.17)$ <p>for kernel k-means;</p> <p><math>u_{k,n} = 1 / \left[ \sum_{k'=1}^K \left( \frac{D_E^k(\Phi(x_n), c_k^\Phi)}{D_E^{k'}(\Phi(x_n), c_{k'}^\Phi)} \right)^{2/(1-m)} \right]</math>, for kernel FCM.</p> <p><b>STEP4:</b> Repeat <b>STEP2</b> and <b>3</b> until <math>\sum_{k=1}^K D_E^k(c_k^\Phi(t), c_k^\Phi(t-1)) &lt; \epsilon</math>, where <math>\epsilon</math> is a small positive number.</p>
---

## 2.2.5 Self Organizing Clustering

The Kohonen self-organising map (SOM) is one of the most popular unsupervised clustering algorithms [51, 100] and has been reviewed in many references [113, 48]. There is another algorithm, called self-organizing oscillator networks (SOON) [91], which also belongs to self-organizing clustering family. The so-called self-organizing means that all the prototypes are attracted to the input patterns in an adaptive fashion. Here, we focus on the SOON algorithm, which makes use of a biological fact that fireflies flash together exhibiting a synchronised firing in groups that physically close to each other. The basic unit of clustering in SOON is an integrate and fire (IF) oscillator representing each object in the dataset.

Suppose that  $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_N\}$  is a set of  $N$  oscillators, where each oscillator  $\mathcal{O}_i$  is characterised by a phase  $\phi_i$  and a state variable  $s_i$ , given by

$$s_i = f_i(\phi_i), i = 1, \dots, K, \quad (2.18)$$

where each function  $f_i : [0, 1] \rightarrow [0, 1]$  is smooth. In [91], the function  $f(\phi)$  was

$$f(\phi) = \frac{1}{b} \ln \left[ 1 + (e^b - 1)\phi \right]. \quad (2.19)$$

whenever  $s_i$  reaches a threshold at  $s_i = 1$ , the  $i$ -th oscillator fires and instantaneously reset to zero, following which the cycle repeats. The firing of all other oscillators  $\mathcal{O}_j (j \neq i)$  can be affected by  $i$ -th oscillator by

$$s_j(t^+) = B(s_j(t) + \epsilon_i(\phi_j)), \quad (2.20)$$

where  $B(\cdot)$  is a limiting function to guarantee that  $s_j(t)$  is confined to  $[0, 1]$ , mathematically expressed by

$$B(s) = \begin{cases} s & \text{if } 0 \leq s \leq 1 \\ 0 & \text{if } s < 0 \\ 1 & \text{if } s > 1 \end{cases} \quad (2.21)$$

The coupling strength of the  $i$ -th oscillator at a given phase  $\phi_j$ ,  $\epsilon_i(\phi_j)$ , is the most important concept of the SOON algorithm, mathematically expressed by

$$\epsilon_i(\phi_j) = \begin{cases} C_E \left[ 1 - \left( \frac{D(\mathcal{O}_i, \mathcal{O}_j)}{\delta_0} \right)^2 \right] & \text{if } D(\mathcal{O}_i, \mathcal{O}_j) \leq \delta_0 \\ -C_I \left[ \left( \frac{D(\mathcal{O}_i, \mathcal{O}_j) - \delta_0}{\delta_1 - \delta_0} \right)^2 \right] & \text{if } \delta_0 < D(\mathcal{O}_i, \mathcal{O}_j) \leq \delta_1 \\ -C_I & \text{otherwise} \end{cases} \quad (2.22)$$

where  $\delta_0$  and  $\delta_1$  are limit distances and  $\delta_1$  is set to be five times  $\delta_0$ .  $C_E$  and  $C_I$  are the maximum excitatory coupling and the maximum inhibitory coupling, respectively. To avoid the need to compute and store the pairwise distances between any pair of objects, prototypes  $\beta = \{\beta_1, \dots, \beta_K\}$  are used to represent clusters of the object in SOON-2, which is summarized in Table 2.4.

Table 2.4: The procedure of SOON-2

<p><b>STEP1:</b>  Initialize a phases <math>\phi_i</math> randomly for <math>i = 1, \dots, N</math>;  Set <math>K</math>, and initialize the Prototypes <math>\beta_k</math> randomly for <math>k = 1, \dots, K</math>;</p> <p><b>STEP2:</b>  Identify the next oscillator to fire, <math>\{\mathcal{O}_i : \phi_i = \max_j \phi_j\}</math>;  Identify the close prototype to the oscillator <math>\mathcal{O}_i \mapsto \beta_k</math>;  Compute <math>D(\beta_k, \mathcal{O}_i)</math> for <math>\forall j \in [1, N]</math>;  Bring <math>\phi_i</math> to threshold, and adjust other phases <math>\phi_j = \phi_j + (1 - \phi_i)</math> for <math>\forall j \in [1, N]</math>;</p> <p><b>STEP3:</b>  <b>for</b> all oscillators <math>\mathcal{O}_j (j \neq i)</math> <b>do</b>      Compute state variable <math>s_j</math> ;      Compute coupling strength <math>\epsilon_i(\phi_j)</math>;      Adjust state variable <math>s_j</math> using (2.20);      Compute compute new phase using <math>\phi_j = f^{-1}(s_j)</math>;  <b>end for</b>  Identify synchronized oscillators and reset their phases;  Update prototype <math>\beta_k</math>;</p> <p><b>STEP4:</b>  Repeat <b>STEP 2 and 3</b> until synchronized group stabilize.</p>
---

### 2.2.6 Self Splitting and Merging Clustering

Self-splitting and merging clustering is an idea in which without setting the number of clusters a priori, the algorithm will converge to a partitioning which reveals the true number of clusters and provides fairly accurate clustering results. Recently, some self splitting-merging clustering algorithms have been developed for both general purpose clustering [120, 59] and gene expression data analysis [111]. A competitive learning paradigm, called one-prototype-take-one-cluster (OPTOC) [120], was proposed in the self-splitting clustering algorithm. There are two advantages of the OPTOC that, firstly, it is not sensitive to initialization, and secondly, in many cases, it is able to find natural clusters. However, its ability to find the natural clusters depends on the determination of suitable threshold, which is difficult [111]. Being aware of the shortcoming of the OPTOC, a self-splitting-merging competitive learning (SSMCL) algorithm [111] based on the OPTOC paradigm was developed for gene expression analysis. The SSMCL initially over-clusters the whole dataset using the OPTOC principle and then merge the groups based on the second order statistical characteristics. However, although the number of clusters can be initially set to any value larger than the number of natural clusters, the SSMCL still needs to set it as close to the number of natural clusters as possible, otherwise, too much computing power will be wasted due to the unnecessary over-clustering and merging. With the similar principle as the SSMCL, over-clustering and merging, a cohesion-based self-merging (CSM) algorithm, which was reported in [59] to combine the k-means and hierarchical clustering, also faces the



same problem of setting the initial number of clusters. Here, we briefly introduce the OPTOC competitive learning paradigm. For each prototype, an online learning vector called asymptotic property vector (APV),  $A_k$ , is assigned to guide the learning the  $k$ -th prototype  $P_k$ . The APV is adapted according to

$$A_k = A_k + \frac{1}{n_A^k} \cdot \delta_k \cdot (x_n - A_k) \Theta(P_k, A_k, x_n) \quad (2.23)$$

where

$$n_A^k = n_A^k + \delta_k \cdot \Theta(P_k, A_k, x_n), \Theta(a, b, c) = \begin{cases} 1 & \text{if } D(a, b) \leq D(a, c) \\ 0 & \text{otherwise} \end{cases} \quad (2.24)$$

and

$$\delta = \left[ \frac{D(P_k, A_k)}{D(P_k, x_n) + D(P_k, A_k)} \right] \quad (2.25)$$

The learning process for  $P_k$  is given by

$$P_k = P_k + \alpha_k \cdot (x_n - P_k) \Theta(P_k, A_k, x_n) \quad (2.26)$$

where

$$\alpha_k = \left[ 1 + \frac{D(P_k, x_n)}{D(P_k, A_k)} \right]^{-2} \quad (2.27)$$

The above OPTOC competitive learning paradigm is an effective technique to implement the self splitting and merging clustering.

### 2.2.7 Ensemble Consensus Clustering

Robustness is one of the desired properties of clustering algorithms, However, there is no perfect method which always gives the best results for all types of datasets. In order to enhance the robustness of clustering, the idea of ensemble consensus clustering has been proposed where the partitioning results of many clustering experiments are combined [97, 73, 34, 98, 5, 103, 106, 6]. These partitioning results may come from different clustering algorithms, or same clustering algorithm with different parameters and initializations, or same clustering algorithm to different re-sampled permutations of the target dataset. Although cluster ensembles have been regarded as promising methods, many obstacles have been found while combining results from different experiments. Due to the fact that clustering is unsupervised, one main problem is that it is not a straightforward task to map a specific cluster from one of the clustering results to its corresponding cluster from another clustering result. Another problem is that different clustering results may give different numbers of clusters while the correct number of clusters is unknown. Consensus function method has been employed as an essential step in cluster ensembles. For  $R$  partitions  $\{U_1, \dots, U_R\}$ , the optimal consensus partition  $U^*$  is the one which is the most similar to all of them and is mathematically given by

$$U^* = \underset{U}{\operatorname{argmax}} \sum_{j=1}^R \Gamma(U, U_j) \quad (2.28)$$

where  $\Gamma(\cdot, \cdot)$  measures the similarity between any two partitions. This optimization problem has been noted as an NP-complete problem. There are many methods for consensus function, including relabelling and voting [6], co-association matrix [37], hyper- graph methods [97], weighted kernel consensus functions [103], non-negative matrix factorization [106], greedy algorithms [34], etc. In all of the aforementioned methods, there are at least three steps to implement the ensemble consensus clustering as follows:

- **Partitions generation:**  $R$  different clustering experiments are carried out to generate  $R$  partitions. The results of these partitions are all presented in a consistent form known as the partition matrix.
- **Relabelling:** The clusters in the generated partitions are relabelled such that the corresponding clusters from different partitions are aligned.
- **Final consensus partition matrix generation:** The relabelled partition matrices are "assembled" to generate the final consensus partition matrix.

Among these three steps, *Relabelling* and *Final consensus partition matrix generation* are the most essential parts. An example of relabelling is detailed in the steps below:

1. A dissimilarity distance matrix  $S_{K \times K}$  is constructed by calculating the pairwise distance between the rows (clusters) of the matrix  $U$  and the rows of the reference matrix  $U^{ref}$ .
2. The minimum value in each of the columns is found.
3. The maximum value of these minima is identified then the rows (clusters) from  $U$  and  $U^{ref}$  which correspond to this similarity value are mapped.
4. The row and the column which show the aforementioned value are deleted from the similarity matrix.
5. If all of the  $K$  rows from  $U$  and  $U^{ref}$  are mapped, the algorithm terminates, otherwise it goes back to step (2) with the reduced similarity matrix.

It is suggested that an intermediate consensus partition matrix  $U^{int(k)}$  is initialized with the values of the first partition  $U^1$ , and then the other partitions are relabelled and fused with this intermediate matrix one by one while considering it as the reference at each step. Mathematically, let  $\hat{U}^r$  be the relabelled partition matrix of the partition  $U^r$  and let  $U^{int(k)}$  be the intermediate partition matrix after the  $k$ -th stage, i.e. after

relabelling and fusing the partitions  $\{U^1, \dots, U^k\}$ . Let the function  $\text{Relabel}(U, U^{ref})$  denote relabelling the partition matrix  $U$  by considering  $U^{ref}$  as the reference partition. Equation 2.14 shows how the intermediate partition matrix can be calculated by the normal approach and the recursive approach:

$$U^{int(k)} = \frac{1}{k} \sum_{r=1}^k \hat{U}^r = \frac{1}{k} \hat{U}^k + \frac{(k-1)}{k} U^{int(k-1)} \quad (2.29)$$

An example of generating the final consensus partition matrix is achieved by following the algorithm shown in the following steps:

1.  $U^{int(1)} = U^1$
2. For  $k = 2$  to  $R$ 
  - a.  $\hat{U}^k = \text{Relabel}(U^k, U^{int(k-1)})$
  - b.  $U^{int(k)} = \frac{1}{k} \hat{U}^k + \frac{(k-1)}{k} U^{int(k-1)}$
3.  $U^* = U^{int(R)}$

## 2.3 Clustering validation

Since clustering is unsupervised classification, it is more difficult to assess than a supervised approach. Thus, the task of assessing the results of clustering algorithms can be as important as the clustering algorithms themselves. There are two functional advantages of employing the clustering validations: firstly, they can validate a clustering algorithm by comparing with other algorithms; secondly, some validity indices can provide an estimate of the number of clusters, which is crucial information for the clustering analysis. In this section, we will review two types of the existing clustering validations including internal evaluating and external evaluating. When internal evaluation is used, the assessment is based on the data matrix itself. The cluster structure possessing higher within-cluster similarity and lower between-cluster similarity is of better quality. When external evaluation is used, the assessment compares the clustering results with the ground truth partition of the dataset. The cluster structure that better matches the ground truth partition is of higher quality.

### 2.3.1 Internal Evaluation

#### Bayesian information criterion index (BIC)

The Bayesian information criterion (BIC) has been proposed in [3] and is defined as:

$$BIC = -\ln(L) + v \ln(n) \quad (2.30)$$

Where  $n$  is the number of objects,  $L$  is the likelihood of the parameters to generate the data in the model, and  $v$  is the number of free parameters in the Gaussian model. Smaller BIC value means better clustering performance.

### Davies-Bouldin index (DB)

This index DB [21] is defined as:

$$BD = \frac{1}{c} \sum_{i=1}^c \text{Max}_{i \leq j} \left[ \frac{d(X_i) + d(X_j)}{d(c_i, c_j)} \right] \quad (2.31)$$

Where  $c$  denotes the number of clusters,  $i, j$  are cluster labels, then  $d(X_i)$  and  $d(X_j)$  are all samples in clusters  $i$  and  $j$  to their respective cluster centroids,  $d(c_i, c_j)$  is the distance between these centroid. Smaller DB value means better clustering performance.

### Parametric Validity Index

A parametric validity index (PVI), which employs two tunable parameters  $\alpha$  and  $\beta$  to control the proportions of objects that are involved in the calculation of the intra-cluster dissimilarities and the inter-cluster dissimilarities, was proposed in [33]. For each cluster, three spaces are defined, namely the inner space, the intra outer space and the inter outer space, representing the objects inside the cluster chosen for the calculation of both the intra-cluster dissimilarities and the inter-cluster dissimilarities, the objects inside the cluster chosen for the calculation of only the intra-cluster dissimilarities, and the objects outside the cluster chosen for the calculation of only the inter-cluster dissimilarities, respectively. Let  $N_k^i$ ,  $N_{a_k^o}$ ,  $N_{e_k^o}$  denote the numbers of objects in the inner space, the intra outer space and the inter outer space, respectively, for the  $k$ th cluster. The fractions,  $\alpha$  and  $\beta$ , are used to control  $N_k^i$ ,  $N_{a_k^o}$ ,  $N_{e_k^o}$ , which can be expressed as

$$N_k^i = \lceil \alpha N_k \rceil; N_{a_k^o} = \lceil \beta N_k \rceil; N_{e_k^o} = \lceil \beta(N - N_k) \rceil; \quad (2.32)$$

where  $N_k$  is the number of the objects in the  $k$ th cluster,  $N$  is the number of all objects in the dataset and  $\lceil \cdot \rceil$  is the ceiling operator. Both  $\alpha$  and  $\beta$  can be chosen from the range of  $(0, 1]$ . Thus, the inner space is  $A_k = \{a_k^a | a = 1, \dots, N_k^i\}$ , the intra outer space is  $B_k^a = \{b_k^{a,b} | b = 1, \dots, N_{a_k^o}\}$ , and the inner outer space is  $C_k^a = \{c_k^{a,c} | c = 1, \dots, N_{e_k^o}\}$ . The PVI is obtained by

$$PVI(K, \alpha, \beta) = \sum_{k=1}^K \sum_{a=1}^{N_k^i} \left( \frac{D_{e_k^a}}{D_{a_k^a}} \right) \quad (2.33)$$

where

$$D_{a_k^a} = \frac{\sum_{b=1}^{N_{a_k^o}} D(a_k^a, b_k^{a,b})}{N_{a_k^o}}; D_{e_k^a} = \frac{\sum_{c=1}^{N_{e_k^o}} D(a_k^a, c_k^{a,c})}{N_{e_k^o}} \quad (2.34)$$

## Other Indices

Here we also list five other existing validity indices. The first is the  $V_I$  [92]. The validity index  $V_I$  is the ratio of the inter-cluster separation measures and the intra-cluster scatter measures, which is mathematically expressed as

$$V_I(K) = \frac{\sum_{i=1}^K I_{e_i}}{\sum_{i=1}^K I_{a_i}} \quad (2.35)$$

where  $K$  is the number of clusters. The  $V_I$  employs  $I_{a_i}$ , the largest dissimilarity of the minimum spanning tree (MST) for cluster  $i$ , as the intra-cluster scatter and  $I_{e_i} = \min_{j=1, j \neq i}^K I_{e_{ij}}$ , where  $I_{e_{ij}}$  is the the largest dissimilarity of the MST for cluster  $i$  and cluster  $j$ , as the inter-cluster separation.

The second is the  $DI$  [29], which is written as

$$DI(K) = \min_{1 \leq i \leq K} [\min_{1 \leq j \leq K} [\frac{\delta(C_i, C_j)}{\max_{1 \leq k \leq K} [\Delta(C_k)]}]] \quad (2.36)$$

where  $\delta(C_i, C_j) = \min \|x_i - x_j\|_2$  is the maximum distance between cluster  $i$  and cluster  $j$ ,  $\Delta(C_k)$  is the largest intra-cluster separation of cluster  $k$ . The third is the  $II$  [70], which is written as

$$II(K) = (\frac{1}{K} \times \frac{E_1}{E_K} \times D_K)^P \quad (2.37)$$

where  $E_1 = \sum_j \|x_j - c\|_2$  where  $c$  is the centroid of the whole dataset,  $E_K = \sum_{k=1}^K \sum_{j \in C_k} \|x_j - c_k\|_2$ ,  $D_K = \max_{i,j}^K \|c_i - c_j\|_2$  and power  $P$  is constant, which is 2 in our experiments. The fourth is the  $GI$  [54], which is expressed as

$$GI(K) = \max_{1 \leq k \leq K} [\frac{(2 \sum_{m=1}^M \sqrt{\lambda_{mk}})^2}{\min_{1 \leq j \leq K} \|u_i - u_j\|_2}] \quad (2.38)$$

where  $M$  is the number of dimensions,  $\lambda_{mk}$  are the eigenvalues of the covariance matrix of the  $k$ -th cluster. Note that the closest  $GI$  value to zero suggests the best number of clusters. The fifth is the  $CH$  [16] which is given by

$$CH(K) = \frac{[\frac{\sum_{k=1}^K n_k \|c_k - u\|^2}{K-1}]}{[\frac{\sum_{k=1}^K \sum_{i=1}^{n_k} \|x_i - c_k\|^2}{n-K}]} \quad (2.39)$$

where  $n_k$  is the number of memberships in the cluster  $k$  and  $n$  is the total number of the objects.

### 2.3.2 External Evaluation

#### RAND

RAND [87] is defined as the probability of correction for the cluster results. It handles two partition matrices defined as  $\mathbf{T}$  and  $\mathbf{Q}$  of the same dataset.  $\mathbf{T}$  encodes the  $k$  known

cluster labels and  $\mathbf{Q}$  records the cluster labels obtained from the algorithms. So the RAND index  $w \in [0, 1]$  is then defined as

$$w = \frac{a + d}{a + b + c + d} \times 100\% \quad (2.40)$$

where  $a$  represents the number of pairs of data points belonging to the same cluster both in  $\mathbf{T}$  and in  $\mathbf{Q}$ ,  $b$  represents the number of pairs of data points belonging to the same cluster in  $\mathbf{T}$  but different clusters in  $\mathbf{Q}$ ,  $c$  represents the number of pairs of data points belonging to different clusters in  $\mathbf{T}$  but the same cluster in  $\mathbf{Q}$ , and  $d$  represents the number of pairs of data points belonging to different clusters both in  $\mathbf{T}$  and in  $\mathbf{Q}$ . Note that a RAND value closer to one suggests the better cluster result.

### Normalized Mutual Information (NMI)

The NMI of two labeled objects can be measured as:

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \quad (2.41)$$

Where,  $I(X, Y)$  denotes the mutual information between two random variables  $X$  and  $Y$  and  $H(X)$  denotes the entropy of  $X$ .  $X$  is clustering result while  $Y$  will be the true labels.

### Purity

The purity of the clustering solution is obtained as:

$$Purity = \sum_{j=1}^m \frac{n_j}{n} P_j \quad (2.42)$$

where  $n_j$  is the size of cluster  $j$ ,  $m$  is the number of clusters and  $n$  is the total number of objects.  $P_j$  is the purity in cluster  $j$  and computed as follows.

$$P_j = \frac{1}{n_j} \text{Max}_i(n_j^i) \quad (2.43)$$

The equation is the number of objects in  $j$  with class label  $i$ .

### Entropy

Entropy measures the purity of the clusters class labels. Thus, if all clusters consist of objects with only a single class label, the entropy is 0. However, as the class labels of objects in a cluster become more varied, the entropy increases. To compute the entropy of a dataset, we need to calculate the class distribution of the objects in each cluster as follows:

$$E_j = \sum_i P_{ij} \log(P_{ij}) \quad (2.44)$$

Where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the weighted sum of the entropies of all clusters, as shown in the next equation.

$$E = \sum_{j=1}^m \frac{n_j}{n} E_j \quad (2.45)$$

Where  $n_j$  is the size of cluster  $j$ ,  $m$  is the number of clusters, and  $n$  is the total number of data points.

## 2.4 Summary

In this chapter, we reviewed some popular clustering algorithms including k-means, k-medoids and HC. Also five families of clustering algorithms are summarised and discussed in one of our conference papers. All these clustering methods here can be used as the clustering based initialisation for NMF. In chapter 4, 5 and 6, we selected k-means and FCM as the examples for NMF initialisation and chose RAND as the measure of the clustering performance. Future works can be done using other clustering methods and clustering validations. In chapter 7, k-means, k-medoids and HC were used to analyse the clustering performance of EEG dataset. Also the ensemble clustering described in section 2.2.7 was applied to get the tight clusters.

## Chapter 3

# Nonnegative Matrix Factorization

This chapter describes the knowledge of Nonnegative matrix factorisation (NMF), emphasis on the topic of both NMF optimization strategies and NMF initialization methods. Section 3.1 provides a brief concept of NMF and its usages. Section 3.2 reviews some NMF optimization strategies. Section 3.3 introduces three types of NMF initialization methods saying randomisation-based initialisation, cluster-based initialization and dimensionality reduction-based initializsation.

### 3.1 Introduction

NMF, proposed by [24], is an algorithm based on decomposition by parts that can reduce the dimensionality of the datasets while keeping the most information about the datasets. It is different from principal component analysis (PCA) and independent component analysis (ICA) with the added non-negative constraints. Researchers have proposed several different algorithms based on the traditional NMF to make improvements such as Least squares-NMF [105], Weighted-NMF [41], Local-NMF [56], and so on. Here we briefly review the idea of NMF as follows. Given a nonnegative matrix  $\mathbf{X} = [x_{ij}]$  with  $m$  rows and  $n$  columns, the NMF algorithm seeks to find nonnegative factors  $\mathbf{W} = [w_{ij}]$  and  $\mathbf{H} = [h_{ij}]$  such that

$$\mathbf{X} \approx \mathbf{WH} \tag{3.1}$$

where  $\mathbf{W}$  is an  $m \times k$  matrix and  $\mathbf{H}$  is a  $k \times n$  matrix. Each column of  $\mathbf{W}$  is considered as the basic vectors while each column of  $\mathbf{H}$  contains the encoding coefficient.  $k$  here is the rank of dimensionality and normally smaller than  $m$  and  $n$  for the aim of the dimensionality reduction. All the elements in  $\mathbf{W}$  and  $\mathbf{H}$  represent non-negative values. NMF method has been found to be useful tool in both data compression and data clustering.



### Data compression

NMF is distinguished from the other dimensionality reduction methods by its non negativity constraints. These constraints can lead to a part-based representation rather than whole-based representation because they only additive rather than subtractive and combinations. This is the good performance in the applicable fields such as image compression, text compression and so on. For example, in the image compression, the figure 3.1 from [23] shows the comparison between NMF basis images and singular value decomposition (SVD) basis images on the face dataset.

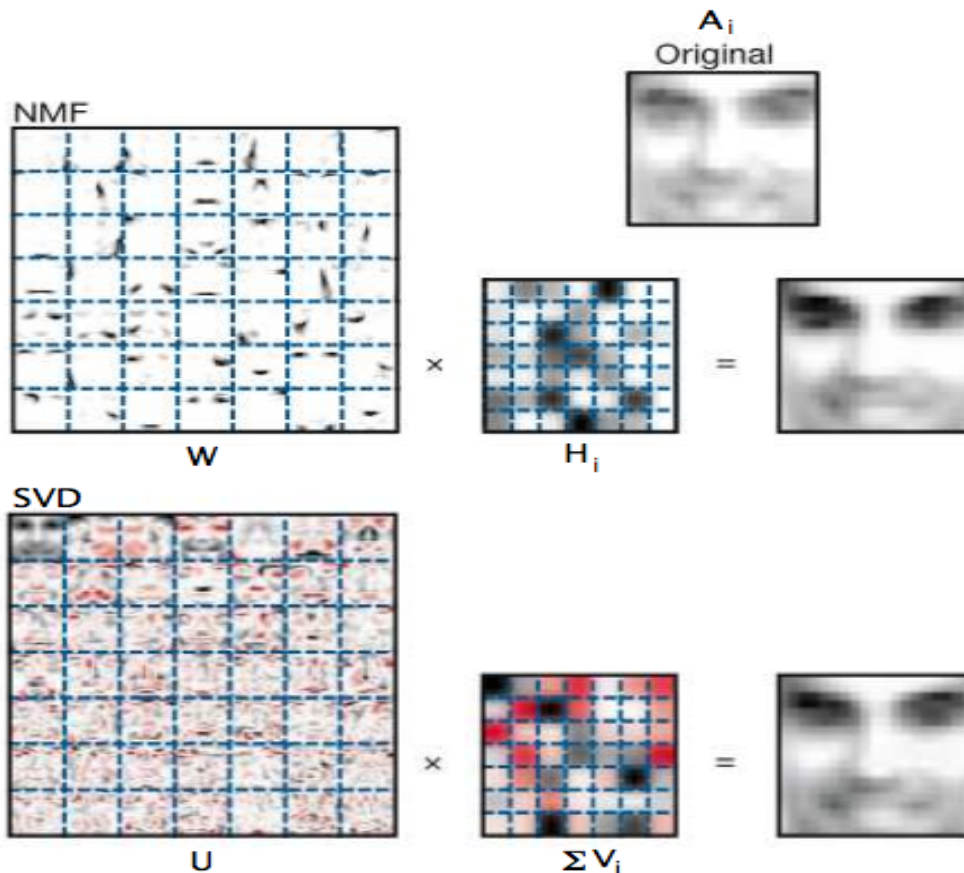


Figure 3.1: Description of NMF and SVD basis vectors on face dataset, from [23]

Both  $W$  and its corresponding weights in  $H$  are sparse while SVD factors are nearly whole-based representation which show the whole faces and therefore needs more computations. NMF basis images  $W$  can be visualised as the parts of faces and have the nice interpretation which shows the individual components of the faces clearly (e.g. ears, noses, mouths and so on). To reach the good compression performance, some properties including sparsity, orthogonality and error are evaluated.

Also NMF can be used for text mining applications. In this process, a document-term matrix is constructed with the weights of various terms (typically weighted word

frequency information) from a set of documents. This matrix is factored into a term-feature and a feature-document matrix.

### Data clustering

NMF is similar to the traditional vector quantisation and k-means clustering [23]. Also recently, the equivalence of NMF and spectral clustering has been proved in [14]. NMF then can be used to facilitate cluster exploration. Brunet et al. compare its clustering performance with self-organizing map (SOM) and hierarchical clustering (HC) [49] and concludes that NMF is an efficient method on identifying the gene expression dataset patterns. After that, a large body of researches has been published to address the analysis extension and application of clustering performance of NMF in image processing, signal processing and data mining during the last decade.

Assume that each column in  $\mathbf{X}$  represents the data points to be analyzed and  $k \leq \min(m, n)$  is often assumed as the number of clusters. Each element of  $\mathbf{H}$  indicates the confidence value a data point belonging a data cluster. The  $i^{th}$  data point is assigned to the  $j^{th}$  cluster when  $j = \operatorname{argmax}_{l=1}^k \mathbf{h}_{li}$ . On the dual view, an element of  $\mathbf{W}$ ,  $\mathbf{W}_{ia} (1 \leq a \leq k)$ , describes the degree of the point  $i$  belonging to the cluster  $a$  and point  $i$  is assigned to cluster  $a$  of  $a = \operatorname{argmax}_{a=1}^k \mathbf{W}_{ia}$  [49]. According to this property of NMF, we can obtain the cluster label for a given dataset from the  $\mathbf{W}$  or  $\mathbf{H}$  value.

## 3.2 NMF optimization strategies

An important group of NMF research is focused on its optimization strategy, studying how to find accurate NMF approximations fast for large data size. Typical NMF approximation approaches are reviewed in [7], for example, alternating least square algorithms [78], gradient descent algorithms [58], and Lee and Seung’s multiplicative update rules based on the creation of an auxiliary function for solving the constrained optimization problem [24, 57]. Among these approaches, the multiplicative update is perhaps the most popular NMF solver. In section 3.2.1, we review the three standard algorithms of multiplicative update rule proposed by Lee et.al. which are based on the different objective functions. In section 3.2.2, we review some other common update rules including alternating non-negative least squares (ANLS) and gradient descent.

### 3.2.1 Multiplicative update rule

The first objective function is commonly used which is the euclidean distance between the dataset  $\mathbf{X}$  and its approximation  $\hat{\mathbf{X}} = \mathbf{WH}$ . Here we use the frobenius norm of the difference  $\|\mathbf{X} - \mathbf{WH}\|_F$  to factorize  $\mathbf{X}$  into the product of these two non-negative matrix above. That is, the objective function (difference) of  $\|\mathbf{X} - \mathbf{WH}\|_F$  is minimized, subject to the constraints of  $\mathbf{W}_{ia} \geq 0$  and  $\mathbf{H}_{a\mu} \geq 0$ , where  $0 \leq i \leq m$ ,  $0 \leq a \leq k$  and

$0 \leq \mu \leq n$ . The details of the difference  $\mathbf{F}_1$  is given as follows.

$$\mathbf{F}_1 = \sqrt{\frac{\sum_{ij} (\mathbf{X}_{ij} - (\mathbf{WH})_{ij})^2}{m \times n}} \quad (3.2)$$

As such, the most popular heuristic solution to the above objective function of NMF is to use the multiplicative update rule.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{(\mathbf{W}^T \mathbf{X})_{a\mu}}{(\mathbf{W}^T \mathbf{WH})_{a\mu}}; \mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{(\mathbf{XH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}} \quad (3.3)$$

where  $(\cdot)^T$  is the transpose operator. Different setups of  $W_0$  and  $H_0$  may lead to the different factorisation results. One traditional strategy is random initialization (RI) that generates elements of  $W_0$  and  $H_0$  in a completely random manner [24].

The second objective function is called the divergence or entropy which is given as follows [24].

$$\mathbf{F}_2 = \sum_{ij} \left( \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{\hat{\mathbf{X}}_{ij}} - \mathbf{X}_{ij} + \hat{\mathbf{X}}_{ij} \right) \quad (3.4)$$

To minimise the  $\mathbf{F}_2$  with respect to the nonnegative factors  $\mathbf{W}$  and  $\mathbf{H}$ , the corresponding multiplicative update rule is introduced [24].

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{\sum_i W_{ia} X_{i\mu} / (WH)_{i\mu}}{\sum_k W_{ka}}; \mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{\sum_\mu H_{a\mu} X_{i\mu} / (WH)_{i\mu}}{\sum_\nu H_{a\nu}} \quad (3.5)$$

The third objective function is called local nonnegative matrix factorisation (LNMF) proposed by Li et al. [56]. It adds some constraints on the divergence algorithm. The details of its objective function are as follows.

$$\mathbf{F}_3 = \sum_{ij} \left( \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{\hat{\mathbf{X}}_{ij}} - \mathbf{X}_{ij} + \hat{\mathbf{X}}_{ij} \right) + \alpha \sum_{ij} u_{ij} - \beta \sum_i v_{ii} \quad (3.6)$$

where  $\alpha$  and  $\beta$  are some positive constants and  $u = W^T W$ ,  $v = H H^T$ . The term " $\alpha \sum_{ij} u_{ij} - \beta \sum_i v_{ii}$ " is added into the objective function in order to minimize the sparsity and orthogonality of the factors. The update rule of the LNMF for  $\mathbf{W}$  is identical to that in equation 3.5 and  $\mathbf{H}$  is

$$\mathbf{H}_{a\mu} \leftarrow \sqrt{\mathbf{H}_{a\mu} \sum_i X_{i\mu} \frac{W_{ia}}{W_{ia} H_{a\mu}}} \quad (3.7)$$

The convergence rate of LNMF is slower than NMF but the achieved factors  $W$  and  $H$  by LNMF are more localised.

### 3.2.2 Other update rules

#### Alternating non-negative least squares (ANLS)

Some successful algorithms are based on alternating non-negative least squares: in each step of such an algorithm, first  $\mathbf{H}$  is fixed and  $\mathbf{W}$  is found by a non-negative least squares solver, then  $\mathbf{W}$  is fixed and  $\mathbf{H}$  is found analogously.

Find  $\mathbf{W}^{k+1}$  such that  $\mathbf{F}_1(\mathbf{W}^{k+1}, \mathbf{H}^k) \leq \mathbf{F}_1(\mathbf{W}^k, \mathbf{H}^k)$  and  
 Find  $\mathbf{H}^{k+1}$  such that  $\mathbf{F}_1(\mathbf{W}^{k+1}, \mathbf{H}^{k+1}) \leq \mathbf{F}_1(\mathbf{W}^{k+1}, \mathbf{H}^k)$

$\mathbf{F}_1$  here is the objective function stated in equation 3.2. The procedures used to solve for  $\mathbf{W}$  and  $\mathbf{H}$  may be the same [18] or different, as some NMF variants regularize one of  $\mathbf{W}$  and  $\mathbf{H}$ . The non-negative least squares solver for  $\mathbf{W}$  is as follows.

$$\mathbf{W}_{ia} = \max \left( 0, \mathbf{X} (\mathbf{H}_{a\mu}^T \mathbf{H}_{a\mu})^{-1} \mathbf{H}_{a\mu}^T \right) \quad (3.8)$$

There are four advantages for ANLS: it is fast, works well in practice, has speedy convergence and only need to initialize  $\mathbf{W}_0$ .

### Gradient descent

The gradient descent is another update rule for NMF to minimise the objective functions. In [18, 17], several gradient-type approaches have been mentioned. The simple additive update for  $\mathbf{H}$  that reduces the euclidean distance is as follows.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} + \eta_{a\mu} [(\mathbf{W}^T \mathbf{X})_{a\mu} - (\mathbf{W}^T \mathbf{W} \mathbf{H})_{a\mu}] \quad (3.9)$$

where

$$\eta_{a\mu} = \frac{\mathbf{H}_{a\mu}}{(\mathbf{W}^T \mathbf{W} \mathbf{H})_{a\mu}} \quad (3.10)$$

As long as  $\eta_{a\mu}$  is sufficiently small, the update rule can converge.

For the divergence  $\mathbf{F}_2$  in the equation 3.4, the gradient descent for  $\mathbf{H}$  is as follows.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} + \eta_{a\mu} \left[ \sum_i \mathbf{W}_{ia} \frac{\mathbf{X}_{i\mu}}{(\mathbf{W} \mathbf{H})_{i\mu}} - \sum_i \mathbf{W}_{ia} \right] \quad (3.11)$$

where

$$\eta_{a\mu} = \frac{\mathbf{H}_{a\mu}}{\sum_i \mathbf{W}_{ia}} \quad (3.12)$$

Again, if  $\eta_{a\mu}$  is small and positive, this update can converge.

## 3.3 NMF Initialization

### 3.3.1 Random initialization

#### Random initialization

The traditional initialization method for NMF is random initialization that generates elements of the initial factors ( $\mathbf{W}_0$  and  $\mathbf{H}_0$ ) in a completely random manner [24]. This method is inexpensive and sometimes provides a good first estimation for NMF algorithm. For example, random initialization can achieve a lower error than the other initialization methods saying PCA-based initialization, fuzzy c-means and Gabor wavelets [122].

### Random acol initialization (RAI)

Instead of forming completely random basis vectors of  $\mathbf{W}_0$ , the RAI method [55] forms each column of  $\mathbf{W}_0$  by averaging  $p$  randomly selected columns of the data matrix  $X$ .

For example,

$$[W_0]_i = \frac{1}{p} \sum_{j \in N_p} [X]_j \quad (3.13)$$

where  $N_p$  denotes a set of  $p$  random integers between 1 and  $n$  which is the number of data points, and  $[\cdot]_i$  denotes the  $i$ -th column of an input matrix. Then,  $\mathbf{H}_0$  is recorded randomly. RAI makes more sense to choose the basis vectors from the given dataset than the random initialization which forms completely random basis vectors. This method is also inexpensive and sometimes may be better than random initialisation.

### Random c initialization (RCI)

This method is similar to random acol initialisation, it forms each column of  $\mathbf{W}_0$  by averaging  $p$  randomly selected columns from the longest columns (2-norm) of the datasets.

$$[W_0]_i = \frac{1}{p} \sum_{j \in N_p} \max(\|[X]_j\|_2) \quad (3.14)$$

## 3.3.2 Cluster-based Initialization

When the output of NMF is used to facilitate cluster exploration, it is natural to conduct the initialisation by linking to a clustering algorithm. In this section, we review the two common cluster-based initialization for NMF saying kmeans-based initialization and fuzzy c-means-based initialization.

### kmeans-based initialisation

k-means based initialization is a member of the clustering-based initialization family for NMF. It selects the cluster result from k-means clustering algorithm as the initial seed for NMF. Here we briefly summarize the process of the k-means based NMF algorithm as follows.

Given the  $m \times n$  nonnegative matrix  $\mathbf{X}$  and each column of  $\mathbf{X}$  represents the data points which need to be clustered.  $\mathbf{A}$  is the transpose of the matrix  $\mathbf{X}$ . In order to avoid the influence of the initial random values of k-means, we apply k-means clustering on the matrix  $\mathbf{A}$  for  $q$  times and select the cluster result from  $q$  times which has the lowest within-cluster sums of point-to-centroid distances. The cluster result includes the  $k$  cluster centroids in the  $k$ -by- $m$  matrix  $\mathbf{C}$  and the cluster labels for the data points (examples) in the  $n$ -by-1 vector  $Idx$ . The initial values  $\mathbf{W}_0$  and  $\mathbf{H}_0$  of NMF then can be described below.

- The initial basis matrix of NMF  $\mathbf{W}_0$  is constructed by transposing the cluster centroids  $\mathbf{C}$  (i.e.  $\mathbf{C}^T$ ).

- The initial matrix of NMF  $\mathbf{H}$  is the  $k \times n$  matrix which can be obtained from  $Idx$  by the following equation.

$$\mathbf{H}_{jl} = \begin{cases} 1 & \text{if } Idx = j \\ 0 & \text{otherwise} \end{cases} \quad (3.15)$$

where  $0 \leq l \leq n$  and  $0 \leq j \leq k$ .

Finally, we apply NMF algorithm to  $\mathbf{X}$  with the initial matrices  $\mathbf{W}_0$  and  $\mathbf{H}_0$  obtained above and calculate the results. The traditional k-means clustering sometimes may not be suitable for a variety of classification application [109] and certainly is not satisfactory to the domain scientists. Some extensions of the k-means initialization have been proposed such as spherical k-means initialization [109]. Wild et.al. take advantage of the spherical k-means because of not only its efficiency and robustness but also the independent centroids. In [115], Yun et.al also have proposed divergence k-means initialization which achieves faster convergence for the face recognition task.

### FCM-based initialisation

Different with k-means clustering, fuzzy clustering is a concept which characterises the membership of each data point in all the clusters with a membership function. The details of FCM algorithm have been discussed in section 2.2.3. We can then obtain the cluster centroids  $C = [c_{ij}]$  and the cluster membership  $U = [u_{ij}]$  from FCM. Similar to the k-means-based initialization, the initial  $\mathbf{W}_0$  is constructed by transposing the cluster centroids  $\mathbf{C}$  (i.e.  $\mathbf{C}^T$ ). The initial  $\mathbf{H}_0$ , proposed by Zheng.et.al [122], is obtained from the membership  $U$  by

$$\mathbf{H}_{ij} = \begin{cases} 1 & \text{if } i = \text{argmax}_i(U_{ij}) \\ 0 & \text{otherwise} \end{cases} \quad (3.16)$$

That is, the maximum value in each column of  $U$  corresponds to 1 in that column of  $\mathbf{H}_0$ .

Figure 3.2 [122] recorded the error (equation 3.2) during the iterations for each NMF initialisation methods. The experiment result shows that the error of FCM-based initialization (i.e. Clustering-based Init) is small and does not change dramatically. However, in the long-term, the random initialization performs better than FCM-based initialization. To overcome this problem, M.Rezaei.et.al have also proposed the novel FCM-based initialisation in which both the initial  $\mathbf{W}_0$  and  $\mathbf{H}_0$  are constructed simultaneously [88]. Here the initial  $\mathbf{H}_0$  is directly equal to the membership  $U$ . This FCM-based initialization method gets the better performance compared with the old FCM initialization [122] and some standard initialisations [88]. As shown in figure 3.3 and 3.4 obtained from [88], the proposed FCM-based initialization outperformed the old FCM-based initialisation as it can get the lower orthogonality, sparsity and error at the final iteration.

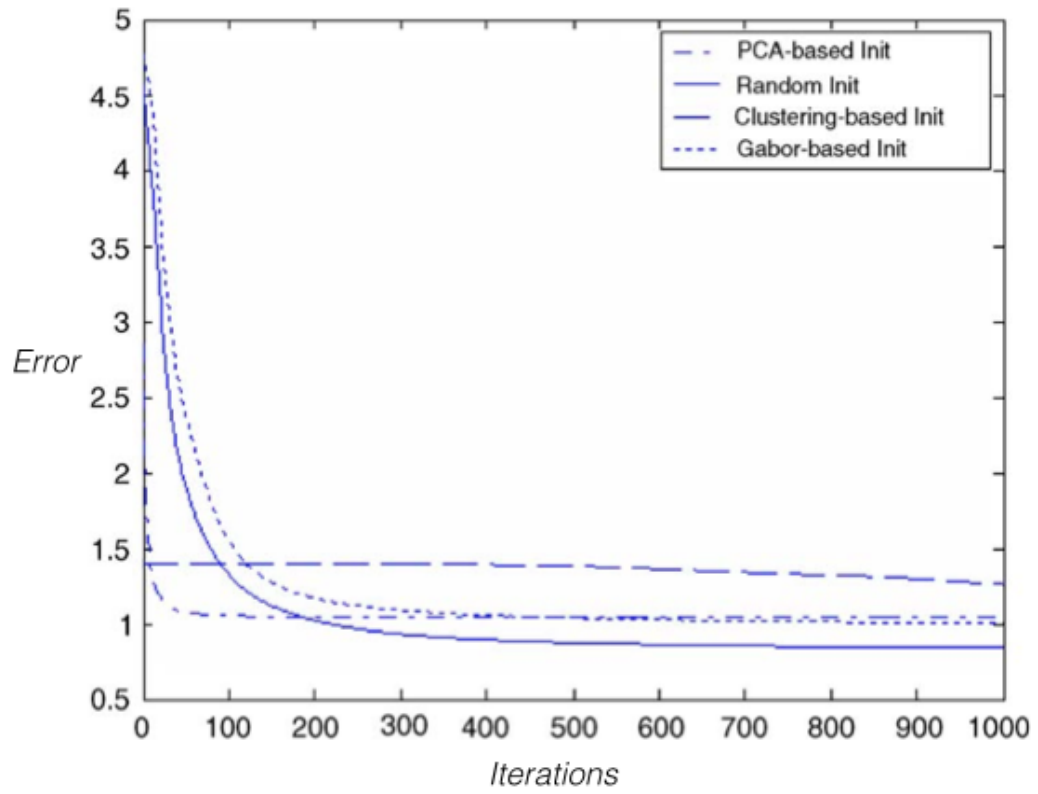


Figure 3.2: Error using different initializations, from [122]

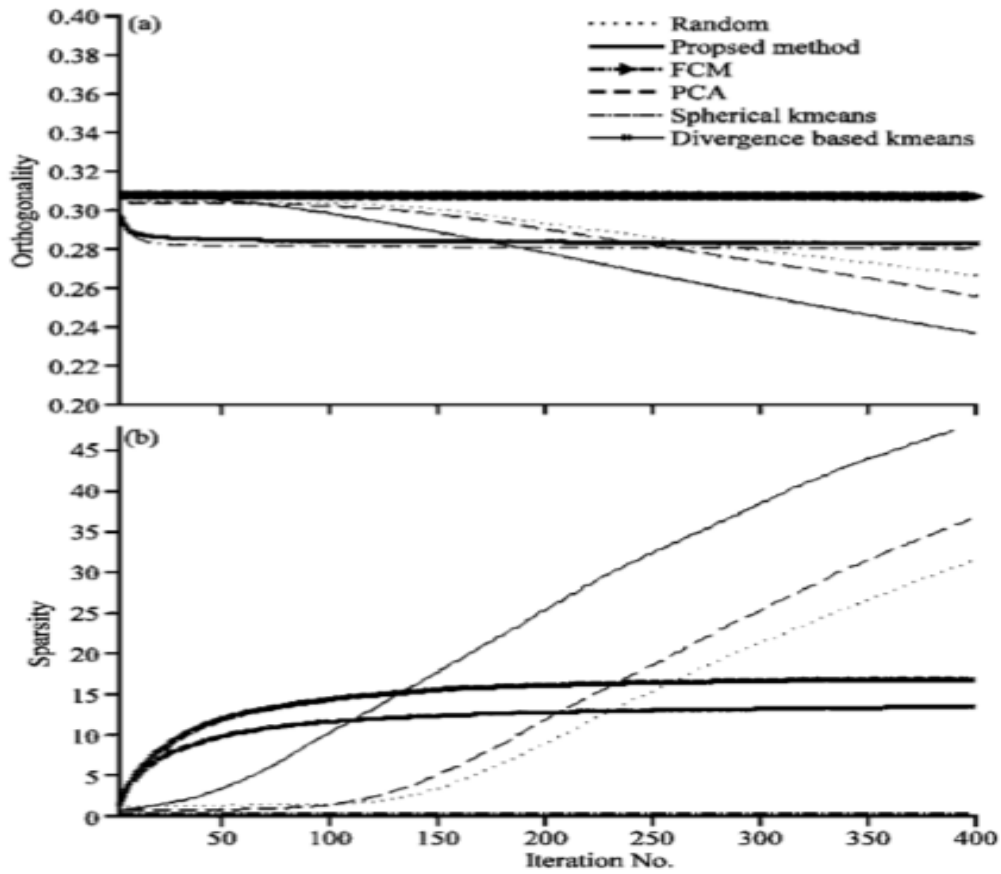


Figure 3.3: Comparison of initialisation methods in terms of sparsity and orthogonality. (a) Comparison of initialisation methods in term of orthogonality and (b) Comparison of initialisation methods in term of sparsity

Initialization methods	Relative error
Random	6.6867
PCA	1.9410
Spherical k-means	0.4520
Divergence-based k-means	0.4488
FCM[5]	0.1922
The proposed method	0.1916

Figure 3.4: Comparison of initialisation methods in terms of error



### 3.3.3 Dimensionality reduction-based initialization

#### 3.3.3.1 Standard Dimensionality Reduction Methods

Besides NMF, there are many other dimensionality reduction algorithms proposed in literatures [35]. With the help of the dimensionality reduction algorithms, high-dimensional dataset can be efficiently brought down to a small number of dimensions without a significant loss of information. The main process of the dimensionality reduction is to keep the most relevant variables from the original dataset and exploit the redundancy of the dataset by finding a smaller set of new variables. Here we introduce the basis idea of three standard dimensionality reduction algorithms saying principal component analysis (PCA), independent component analysis (ICA) and independent principal component analysis (IPCA).

#### Principal Component Analysis (PCA)

PCA is one of the dimensionality reduction method in which the low-dimensional representation of the dataset preserves as much of its variation as possible to highlight its similarities and differences. Since patterns in datasets can not be easily read in terms of the high dimensions, PCA is also a powerful tool for analyzing dataset clearly. It can compress the dataset by removing the useless information. This technique has been used in many fields such as data compression, image compression and so on. Also the pattern of the dataset can be visualisable when the dimensions are reduced to 1D, 2D or 3D. Here we review the steps for PCA in details.

#### STEP1: Dataset for PCA

Given a  $n \times d$  matrix  $X = [x_{ij}]$ , each row represents the corresponding data point and each column represents the dimension to be reduced by PCA.

#### STEP2: Mean subtracted from the dataset

In order to reduce the complexity, the mean obtained from each of the columns (dimensions) is subtracted from the elements in  $X$

$$[X_{adjust}]_i = [X]_i - [\hat{X}]_i \quad (3.17)$$

where  $[\hat{X}]_i$  is the mean across  $i$ -th column (dimension).  $[X]_i$  is the original dataset at the  $i$ -th column. The adjusted dataset  $X_{adjust}$  is the matrix with the zero mean.

#### STEP3: Obtain the covariance matrix of $X_{adjust}$

The formula for the computation of the coveriance between  $[X_{adjust}]_{i_1}$  and  $[X_{adjust}]_{i_2}$  is as follows.

$$cov([X_{adjust}]_{i_1}, [X_{adjust}]_{i_2}) = \frac{\sum_{j=1}^n [X_{adjust}]_{i_1 j} [X_{adjust}]_{i_2 j}}{n - 1} \quad (3.18)$$

where  $n$  is the number of data points.  $[X_{adjust}]_{i_1 j}$  is the value at  $i_1 - th$  column and  $j - th$  data point in  $X_{adjust}$ . If the value  $cov(\cdot)$  is positive, it indicates that both dimensions  $[X_{adjust}]_{i_1}$  and  $[X_{adjust}]_{i_2}$  increase together. If the value is negative, then as one dimension increases, the other decreases. And if the covariance is zero, it indicates that the two dimensions are independent of each other. The covariance in equation 3.18 is measured between any two dimensions (columns). For the dataset with more than two dimensions, the  $d \times d$  covariance matrix  $CM = [cm]_{ij}$  is then calculated by putting all the covariance values between all the different dimensions together. The formula of the covariance matrix is as follows.

$$CM^{d \times d} = [cm]_{ij} = cov([X_{adjust}]_i, [X_{adjust}]_j) \quad (3.19)$$

where  $d$  is the total number of column in dataset and there are totally  $\frac{d!}{(n-2)! \times 2}$  covariance values in the  $CM$ . Because of  $cov(x, y) = cov(y, x)$  according to the equation 3.18, the covariance matrix  $CM$  is symmetrical by the diagonal.

**STEP4:** Obtain the eigenvectors and eigenvalues of the covariance matrix

As mentioned before, the covariance matrix  $CM$  is always symmetric, SVD can be used to find its eigenvalues and eigenvectors [85]. So we have  $d$  normalised eigenvectors:

$$e_k = [e_{1k}; e_{2k}; \dots; e_{dk}], k = 1, \dots, d \quad (3.20)$$

and their corresponding  $d$  eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \quad (3.21)$$

Here  $d$  eigenvectors  $e_k$  are perpendicular to each other and sorted in the decreasing order by their corresponding eigenvalues. The first eigenvector  $e_1$  which has the largest eigenvalue identifies the direction with the most variation in the dataset. The second eigenvector  $e_2$  is perpendicular to the first one and captures the direction with the greatest remaining variation. The third eigenvector  $e_3$  is perpendicular to both  $e_1$  and  $e_2$  and still captures the direction with the greatest remaining variation and so on. So with the help of the eigenvectors  $\hat{E} = (e_1, e_2, \dots, e_d)$  and eigenvalues of the covariance matrix, we can extract the lines that characterise the dataset.

**STEP5:** The selection of the dimensionality

One main advantage of PCA is that it can compress the dataset without much loss of information. Here some eigenvectors with lesser significance can be removed and the significance of the  $k - th$  eigenvector  $e_k$  can be evaluated as follows.

$$\gamma_k = \frac{\lambda_k}{\sum_{i=1}^d \lambda_i}; k = 1, 2, \dots, d \quad (3.22)$$

Assume that the dimension is reduced to be  $m(m \leq d)$  (i.e. the first  $m$  eigenvectors are kept as  $E = [e_1, e_2, \dots, e_m]$ ), the proportion of the total variance of these eigenvectors are

$$\hat{\gamma}_m = \sum_{h=1}^m \gamma_h = \frac{\sum_{h=1}^m \lambda_h}{\sum_{i=1}^d \lambda_i} \quad (3.23)$$

Then the most significant eigenvectors can be kept and the final dataset will have less dimensions than the original dataset. For visualisation, normally two or three eigenvectors are retained.

**STEP6:** Calculate the new dataset in terms of the eigenvectors

$$X_{new} = E^T [X_{adjust}]^T \quad (3.24)$$

$(\cdot)^T$  is the transpose of the matrix and each column of  $X_{new}$  represents the data point.  $X_{new}$  we obtained here is the new dataset in terms of our selected eigenvectors and it has reduced the dimensionality.

The principal components found in PCA are uncorrelated variables constructed as linear combinations of the original variables and have some desirable properties. But sometimes that do not necessarily correspond to the meaningful information and may lose the interpretation.

### Independent Component Analysis (ICA)

ICA is another dimensionality reduction method in which the goal is to find a linear representation of non-gaussian signal so that the components are statistically independent. ICA can be treated as the method to remove most of the noise from the signal (when the noise has a Gaussian distribution). Similar to PCA, ICA [15, 90, 50, 43] will find the new components in which to represent the dataset. However, different with PCA, the components in ICA are chosen to be as statistically independent as possible. Several algorithms such as FastICA [1, 2], kernel-ICA [32] and ProDenICA [99] were proposed to realise the independent components. Among them, FastICA is the most popular ICA algorithm which maximises the non-gaussianity to recover all ICs with low computational load. Here we review the basis knowledge of ICA. Given the  $n \times p$  matrix  $X = [x]_{ij}$  which is a linear mixture of the independent components, saying

$$X = AS \quad (3.25)$$

where  $S \in \mathbb{R}^{n \times p}$  contains the independent components and  $A$  is the mixing matrix with  $n \times n$  size. The matrix  $A$  states the relationship between the independent components  $S$  and the mixture signal  $X$ . In order to find the estimated  $n \times p$  independent components  $Y = [y]_{ij}$  by the recorded matrix  $X$  using ICA, the equation 3.25 is re-arranged as follows.

$$Y = WX \quad (3.26)$$

where  $W \in \mathbb{R}^{n \times n}$  is the unmixing matrix for which ICA is seeking. The matrix  $W$  is the transpose of  $A$  when  $A$  is a square and orthonormal matrix. In the FastICA algorithm, PCA is used to pre-whiten the dataset to make the matrix  $A$  to be orthogonal.

### Independent Principal Component Analysis (IPCA)

Sometimes, PCA and ICA may not be able to extract the significant information from the internal structure of the dataset and therefore provide meaningless components that do not correctly describe the dataset characteristics. Yao.et.al. [118] recently proposed an approach called IPCA which combines the advantage of both PCA and ICA. ICA used in IPCA is a de-noising process of the eigenvectors produced by PCA. Once the eigenvectors is denoised, we expect it to be non-gaussianity with no noises included. The details of IPCA is reviewed as follows [118].

**STEP1:** Calculate the eigenvectors from PCA

Given the  $n \times p$  matrix  $X$ , PCA is then applied to it to obtain the  $p \times p$  matrix  $\hat{E} = [e_1, e_2, \dots, e_p]$  as described in PCA method. Each column of  $\hat{E}$  contains one eigenvector and in most of the cases, a small number of eigenvectors is chosen to summarise the information of the dataset. Assume that the first  $m$  eigenvectors  $E = [e_1, e_2, \dots, e_m], E \in \mathbb{R}^{p \times m}$  are chosen at the first step.

**STEP2:** Apply ICA on the eigenvectors

The non-gaussianity of  $E$  can be maximised by FastICA using equation 3.26 in ICA method

$$S = WE \tag{3.27}$$

where  $W$  is the  $p \times p$  unmixing matrix. The estimated independent components  $S$  with  $p$  rows and  $m$  columns is then obtained. Each column of  $S$  represents the independent eigenvectors which has the less noise than  $E$ .

**STEP3:** Order the independent eigenvectors  $S$

Different with PCA, the independent components after ICA is unordered. To solve this problem, there are two classical measurements for ordering, saying kurtosis and negentropy. In the fastica algorithm, negentropy is used to measure non-gaussianity. Negentropy equals zero if the estimated independent components are gaussian and is positive if the independent components are non-gaussian. Here the kurtosis measurement [65] is chosen to order the independent eigenvectors  $S$ . All the kurtosis values of the independent eigenvectors in  $S$  are calculated

$$\mathbf{Kurtosis} = \frac{\tilde{s}^4}{\sigma^4} - 3 \tag{3.28}$$

and ordered in the decreasing way to form the ordered independent eigenvectors  $\hat{S}$

**STEP4:** Calculate the new dataset in terms of the independent eigenvectors.

$$X_{new} = \hat{S}^T [X_{adjust}]^T \quad (3.29)$$

where  $X_{new}$  is a  $m \times n$  matrix.

As analysed in [118], it showed that IPCA offers a better visualisation of the datasets than ICA and with a smaller number of components than PCA. For example, as shown in figure 3.5 [118], it seems that the first independent principal component of IPCA has already separate the groups while three principal components are necessary for PCA to separate "normal" from "tumour" on prostate cancer dataset.

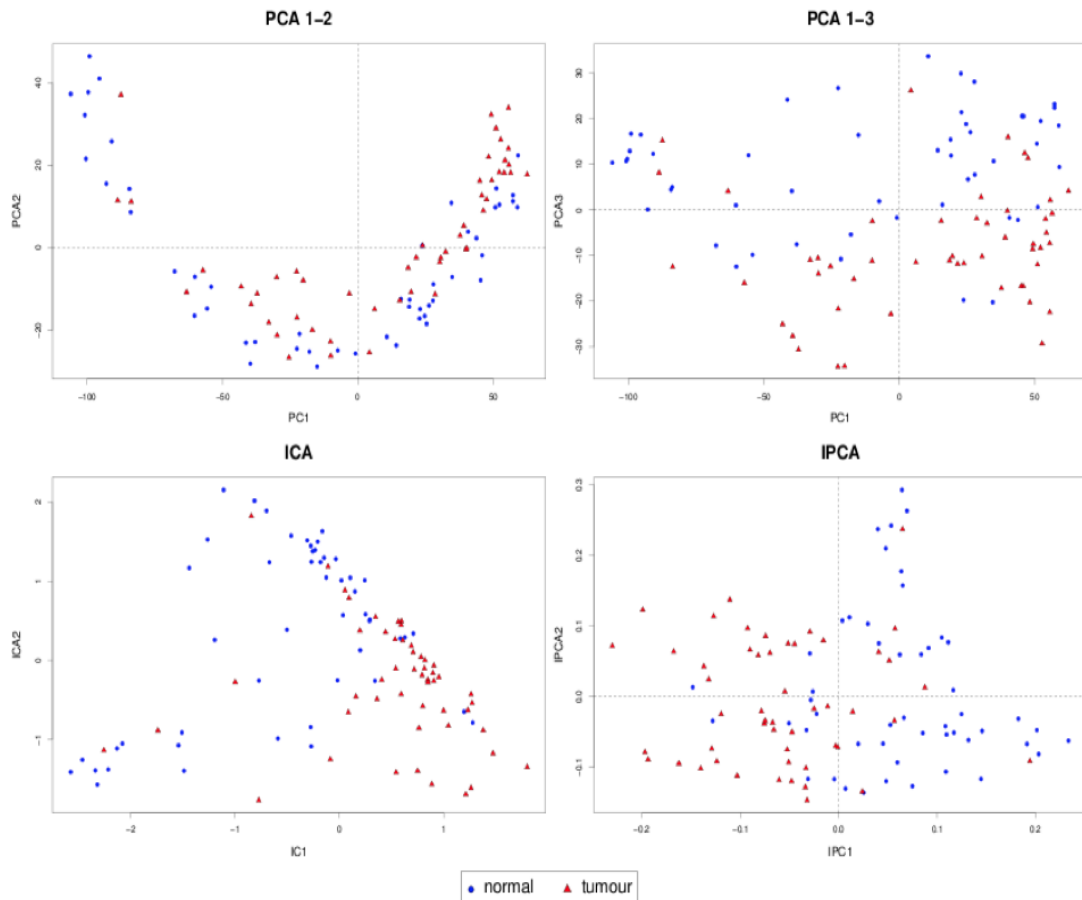


Figure 3.5: Prostate cancer study: sample representation using the first two or three components from PCA, ICA and IPCA, from [118]

### 3.3.3.2 PCA and ICA based Initialization

#### PCA-based initialisation

PCA as mentioned in section 3.3.3.1 is one of the common dimensionality reduction methods and it also has been proved in [122] that it can be used as the initialization

method for NMF. Here we review the PCA based initialization method for NMF as follows [122].

**STEP1:** Given a  $n \times d$  matrix  $X = [x]_{ij}$  and its transpose  $A = [a]_{ij}$

**STEP2:** We apply PCA to the matrix  $A$  and obtain the  $n$  eigenvectors and their corresponding eigenvalues  $\hat{E} = [e_1, e_2, \dots, e_n]$  and  $\lambda_1 > \lambda_2 > \dots > \lambda_n$ .

**STEP3:** Assume the rank of dimensionality for NMF is  $k$ , then the initial matrix  $\mathbf{W}_0$  of NMF is constructed by keeping the first  $k$  eigenvectors (corresponding to the  $k$  largest eigenvalues) obtained from PCA as the column vectors. (i.e.  $W = [e_1, e_2, \dots, e_k]$ )

**STEP4:** The initial  $\mathbf{H}_0$  is formed by the new dataset in terms of the selected eigenvectors  $\mathbf{W}_0$ .

$$H_0 = W^T[A_{adjust}]^T \quad (3.30)$$

where  $A_{adjust}$  is the centred  $d \times n$  matrix (the mean of each column has been subtracted).

**STEP5:** Obtain the nonnegative values of  $\mathbf{W}_0$  and  $\mathbf{H}_0$

The sign of  $\mathbf{W}_0$  and  $\mathbf{H}_0$  from STEP3 and STEP4 is arbitrary. In order to satisfy the nonnegative constraint of NMF, all the negative elements in  $\mathbf{W}_0$  and  $\mathbf{H}_0$  are converted to be zero.

$$W_+ = \max(0, w_{ij}); H_+ = \max(0, h_{ij}) \quad (3.31)$$

where  $w_{ij}$  and  $h_{ij}$  are the elements in  $\mathbf{W}_0$  and  $\mathbf{H}_0$  respectively.

**STEP6:** Apply NMF with the initial values  $W_+$  and  $H_+$

The reason for using the PCA-based initialization is that PCA gives a proper approximation to the matrix factorization. Also the equation 3.31 enhances the sparsity of  $\mathbf{W}_0$  and  $\mathbf{H}_0$  which can reduce the computational complexity. Zheng.et.al have showed that PCA-based initialization has the slightly higher error than random initialisation in the long-term [122]. But its basis images are more sparse and orthogonal. Similar with the PCA-based initialisation, ICA-based initialization selects the independent components as the initial  $\mathbf{W}_0$  and the projections ('coefficients') as the initial  $\mathbf{H}_0$ .

### ICA-based initialization:

ICA is also one of the initialisation methods for NMF. However, before initialization, the components of ICA need to be considered. Different with PCA, ICA does not order its components by 'relevance'. Therefore, some authors proposed to order them either with respect to their kurtosis values, or with respect to their  $I_2$  norm, or by

using Bayesian frameworks to select the number of components. Here we compare the kurtosis values of the independent components  $\mathbf{S}$  obtained from ICA to order the ICs. The formula is:

$$\mathbf{K} = \frac{\tilde{s}^4}{\sigma^4} - 3 \quad (3.32)$$

where  $\tilde{s}$  is the centered vector (the mean of each column has been subtracted) for one of the independent components  $s$  and  $\sigma$  is the standard deviation for  $s$ . The initial factors  $\mathbf{W}_0$  and  $\mathbf{H}_0$  of NMF based on ICA initialization then can be described below.

- The initial matrix  $\mathbf{W}$  of NMF is constructed by keeping the first  $k$  ordered independent components (corresponding to the  $k$  largest kurtosis values) as column vectors.
- The initial matrix  $\mathbf{H}$  of NMF is computed by  $\mathbf{H} = \mathbf{W} \setminus \mathbf{X}$ .

With this basis concept, other dimensionality reduction based initialisation methods have been proposed such as SVD-based initialisation [55] and NNDSVD [12] and so on.

### 3.3.4 Why initialization

NMF is non convex programming in the iteration process, thus it may lead to the different results with the different initial values of  $\mathbf{W}_0$  and  $\mathbf{H}_0$ . A good initialization method can improve the speed and accuracy of NMF, as it can produce faster convergence to an improved local minimum [95]. In this section, we compare two initialisation procedures (random and random acol initialisations) by testing them on the multipliative update algorithm presented in section 3.2.1 to illustrate the importance of NMF initialisation. Given a  $3 \times 3$  matrix as follows:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

#### Random initialisation

The final matrices  $\mathbf{W}$  under random initialisation after 500 iterations is:

$$\begin{pmatrix} 3.5 & 0.4 \\ 6.3 & 3.3 \\ 9.1 & 6.3 \end{pmatrix}$$

and the final  $\mathbf{H}$  is:

$$\begin{pmatrix} 0.2 & 0.5 & 0.8 \\ 0.8 & 0.5 & 0.2 \end{pmatrix}$$

#### Random acol initialisation

The final matrices  $\mathbf{W}$  under random initialisation after 500 iterations is:

$$\begin{pmatrix} 0.7 & 3.2 \\ 4.5 & 4.9 \\ 8.3 & 6.7 \end{pmatrix}$$

and the final  $\mathbf{H}$  is:

$$\begin{pmatrix} 0.7 & 0.6 & 0.4 \\ 0.1 & 0.5 & 0.9 \end{pmatrix}$$

The table below shows the comparison between random and random acol initialisation. Here the error was calculated by equation 3.2 and the sparsity of the matrix  $\mathbf{W}$  was obtained according to the equation 6.4. Although the final errors are the same, the initial error of random acol initialisation is much smaller than that of random initialisation. Also the lower sparsity means the lower computational complexity. So the table 3.1 illustrated that the NMF with random acol initialisation has the better performance than the NMF with no initialisation. This is the simple example to illustrate the importance of NMF initialisation.

Table 3.1: The comparison between random and random acol initialization

Intializations	Intial error	Final error	Sparsity ( $\mathbf{W}$ )
random	0.8	0	14.5
random acol	0.4	0	14.2

### 3.4 Summary

This chapter reviews the knowledge of Nonnegative matrix factorisation, emphasis on the topic of both NMF optimization strategies and NMF initialization methods. We reviewed the NMF optimization strategies including multiplicative update rule, alternating nonnegative least squares and gradient descent. Also we reviewed several standard initialization methods for NMF based on the randomisation, clustering algorithms and the dimensionality reduction methods. The reason for reviewing is that our proposed research works are based on the NMF initialisation and optimization methods. We proposed the novelty initialisation methods and optimization strategy for NMF to improve its performance. The details of the proposed methods are introduced in the following chapters.



## Chapter 4

# Proposed NMF Initialization Strategy

This chapter describes two proposed initialization methods for NMF which are based on clustering and dimensionality reduction algorithms. Section 4.1 introduces the motivation and contribution summary of our proposed initialisation methods. Section 4.2 and 4.3 introduce the details of two proposed initialisation methods respectively. Works of this chapter are published in:

Liyun Gong and Asoke K. Nandi, Clustering by Non-negative Matrix Factorization with Independent Principal Component Initialization. European Signal Processing Conference, EUSIPCO, 2013. (acceptance rate=62.5%)

Liyun Gong and Asoke K. Nandi, An Enhanced Initialization Method for Non-negative Matrix Factorization, IEEE International Workshop on Machine Learning for Signal Processing, MLSP, 2013. (acceptance rate=51%)

### 4.1 Introduction

Non-negative matrix factorization (NMF) is one of the widely used tools for dimensionality reduction, and has been applied to many areas such as bioinformatics, face image classification, etc. However, it sometimes converges to some local optima because of its random initial NMF factors ( $\mathbf{W}$  and  $\mathbf{H}$  matrices). To solve this problem, some researchers have paid much attention to the NMF initialization problem. Wild proposed the initialization method based on spherical k-means clustering to initialize  $\mathbf{W}$  matrix and apply Nonnegative Least Square (NNLS) to calculate  $\mathbf{H}$  matrix [109]. Langville et al. compared the six initialization methods, including random initialization, centroid initialization, SVD-centroid initialization, random acol initialization, random C initialization, and co-occurrence initialization [55]. Zheng et al. proposed PCA-based initialization method and, after obtaining  $\mathbf{W}$  and  $\mathbf{H}$ , all the negative elements in these two matrices are changed to zero [122]. For fuzzy c-means clustering (FCM) initialization in [122], the initial matrix  $\mathbf{W}$  is obtained by taking the final cluster centroids after

FCM algorithm and the initial matrix  $\mathbf{H}$  is determined by the membership degrees of FCM in which the largest membership degree of each data point is set to one and the others to zero. This method was followed by Rezaei et al. who used the membership degrees of FCM to determine the initial  $\mathbf{H}$  matrix directly [88]. Zhao et al. used the absolute value for all elements in the initial matrices  $\mathbf{W}$  and  $\mathbf{H}$  after PCA initialization [121] in contrast to the method in [122]. Boutsidis et al. proposed the initialization method based on singular value decomposition [12]. There are other initialization methods such as divergence-based k-means clustering initialization, subtractive clustering initialization and ICA-based initialization and so on. With these initialization methods, enhanced convergence rates as well as better accuracy can be achieved. Here we also proposed two initialization methods for NMF which get the better performance compared with some standard methods.

In section 4.2, we propose an initialization methods for NMF based on the modified k-means clustering and explore the reconstructive error of the initialised NMF. Since there are two ways for the FCM-based initialization methods mentioned in section 3.3.2, k-means initialisation methods can have the different types as well. Here we first apply the k-means clustering to initialize the factor  $\mathbf{W}_0$ , and then we calculate the initial factor  $\mathbf{H}_0$  using four different initialization methods (three standard and one new). The experiments were carried out on the eight real datasets and the results showed that the proposed method (EIn-NMF) achieved less error and faster convergence compared with both random initialization based NMF and the three standard methods for k-means based NMF.

In section 4.3, we propose an initialization method based on independent principal component analysis (IPCA) for NMF and explore the clustering performance of the initialized NMF. As there are some papers for PCA-based and ICA-based initialisations [122, 119], IPCA-based initialisation for NMF is the new concept in this area. In the conference, the experiments were carried out on the four real datasets and the results showed that the IPCA-based initialization of NMF gets better clustering of the datasets compared with both random and PCA-based initializations. After this publication, we made some improvements on this conference paper by adding another five datasets and IPCA-based initialization is compared with ICA-based initialisation as well. The results show the IPCA-based initialization still can get the best clustering performance among the standard methods.

## 4.2 Clustering Based Initialization

In this section, we propose a method to calculate the nonnegative initial matrix  $\mathbf{H}$  efficiently in order to enhance the performance of NMF algorithm based on k-means clustering initialization. We first apply k-means clustering to initialize the matrix  $\mathbf{W}$  [109] and use the proposed method to initialize the  $\mathbf{H}$  matrix (EIn-NMF). These initial

matrices are then passed through NMF algorithm and the result will be compared with both random initialization based NMF and the three standard methods for k-means based NMF. The experiments were carried out on the eight real datasets from [61] and the results showed that the proposed method (EIn-NMF) achieved less error and faster convergence while maintaining the main data structure.

The rest of the section is organized as follows. In Section 4.2.1 we describe the details of the proposed initialization method (EIn-NMF). The experimental results based on the five initialization methods (four standards and one new) are evaluated and analysed in Section 4.2.2. Finally, conclusion is drawn in Section 4.2.3.

**Contribution of this section:** we propose a new initialization method (EIn-NMF) for NMF to improve its performance. We also compare four standard initialization methods which have not been compared and contrasted in the previous literatures.

#### 4.2.1 Method

There are two ways for FCM-based initialisation introduced in section 3.3.2. One uses FCM for computing matrix  $\mathbf{H}$ , largest membership degree of each sample is set to one and others assign to zero. The other one uses FCM for calculating matrix  $\mathbf{H}$ , the membership degrees of each sample directly. As FCM-based initialisation has two ways for computing the initial  $\mathbf{H}$ , k-means initialisation can also have the different types as well. Here we propose to use the k-means clustering initialization of  $\mathbf{W}$  incorporated with our proposed initialization of  $\mathbf{H}$  to improve the performance of k-means based NMF. This initial  $\mathbf{H}$  is obtained by computing the FCM membership function for each sample so that each sample can assign to all clusters which increases the clustering accuracy as well as the accuracy of  $\mathbf{H}$ . The details of this method are shown below.

**Proposed:** The initial basis matrix  $\mathbf{W}$  is constructed by using the cluster centroids obtained from k-means clustering as described in section 3.3.2. Then we calculate the membership degrees of each data point by the following equation.

$$h_{kq} = 1 / \left[ \sum_{k'=1}^k \left( \frac{d(x_q, c_{k'})}{d(x_q, c_k)} \right)^{2/(1-m)} \right] \quad (4.1)$$

where  $d(\cdot)$  represents the Euclidean distance between the two points,  $x_q$  represents the  $q$ -th data point and  $c_k$  represents the  $k$ -th cluster centroid which is obtained from k-means. Also,  $m$  is the fuzzification parameter which is set to 2 here.  $h_{kq}$  is the membership degree of data point  $x_q$  assigning to the cluster  $k$ . The initial matrix  $\mathbf{H}$  is then obtained by using the membership degrees above. This proposed enhanced initialization method for NMF is termed as EIn-NMF.

#### 4.2.2 Results

Eight datasets used here are from [61] and they are gene expression datasets. Columns of the datasets corresponding to diagnosis (1st column) and genes, and rows correspond-

ing to samples. Based on the similar preprocessing method as in [61], we removed the genes (columns) which vary little across samples (rows) to reduce the computational complexity. Table 4.1 shows some parameters for preprocessing. 'N' represents no operation performed and 'inf' represents the infinite value. 'Ratio', 'Diff.' and 'Std.' are the ratio, difference and standard values between maximum and minimum values of a gene (column) across samples (rows). We first set all the expression levels within the defined range. For each dataset, elements below the minimum value of the range were assigned this minimum value, and those exceeding the maximum value of the range were assigned this maximum value. Then some genes varying little across samples were removed according to the three parameters (Ratio, Difference and Standard), that is, we remove the columns which have the smaller values on Ratio, Difference and Standard. After preprocessing, the properties of these dataset are presented in Table 4.2. For simplicity, we use the 'Dataset No.' shown in Table 4.2 instead of its gene name for the following discussion.

Table 4.1: The parameters for the preprocessing

Name	Range	Ratio	Diff.	Std.
9-Tumors	[100,16000]	<i>N</i>	<i>N</i>	<i>N</i>
Brain-Tumors1	[20, 16000]	3	500	<i>N</i>
Brain-Tumors2	[20, 16000]	3	500	<i>N</i>
Leukemia1	[20, 16000]	3	500	<i>N</i>
Lung-Cancer	[0, <i>inf</i> ]	<i>N</i>	<i>N</i>	50
SRBCT	<i>N</i>	<i>N</i>	<i>N</i>	<i>N</i>
Prostate-Tumor	[20, 16000]	5	<i>N</i>	<i>N</i>
DLBCL	[20, 16000]	3	500	<i>N</i>

Table 4.2: The properties of the final datasets

Dataset No.	Name	Rows	Columns	Cluster
1	9-Tumors	60	5726	9
2	Brain-Tumors1	90	4922	5
3	Brain-Tumors2	50	1013	4
4	Leukemia1	72	3621	3
5	Lung-Cancer	203	3312	5
6	SRBCT	83	2308	4
7	Prostate-Tumor	102	3722	2
8	DLBCL	77	4204	2

The experiments were carried out by using above eight datasets, we applied five different initialization methods denoted by ( $\mathbf{H}_1 - \mathbf{H}_5$ ) to improve the performance of NMF.  $\mathbf{H}_5$  represents the proposed methods. ( $\mathbf{H}_1 - \mathbf{H}_4$ ) represent four standard initialisations stated below.

- $\mathbf{H}_1$ : This is the common method for initializing both the two NMF factors  $\mathbf{W}$

and  $\mathbf{H}$  randomly.

- $\mathbf{H}_2$ : The initial basis matrix  $\mathbf{W}$  is constructed by using the cluster centroids obtained from k-means clustering and the initial matrix  $\mathbf{H}$  is selected randomly.
- $\mathbf{H}_3$ : The initial basis matrix  $\mathbf{W}$  is constructed by using the cluster centroids obtained from k-means clustering. The initial matrix  $\mathbf{H}$  is denoted by  $\mathbf{H} = \mathbf{W}^T \mathbf{X}$  and then we get the absolute value for all elements in  $\mathbf{H}$  in order to satisfy the initial constraint of NMF.
- $\mathbf{H}_4$ : The initial basis matrix  $\mathbf{W}$  is constructed by using the cluster centroids obtained from k-means clustering. The initial matrix  $\mathbf{H}$  is denoted by  $\mathbf{H} = \mathbf{W}^T \mathbf{X}$  and then we set all the negative elements in  $\mathbf{H}$  to zero in order to satisfy the initial constraint of NMF.

Each initialization method was run 20 times and the total number of iterations for each run of NMF was set to 500 in this experiment. Here we calculate the range of the standard deviation among the total 500 iterations after 20 rounds in table 4.3. It shows that the standard deviation values are all small enough so that 20 rounds are sufficient to avoid the influence of randomness. The rank (dimensionality)  $k$  for each dataset here is set to the number of cluster of the corresponding dataset which is shown in Table 4.2.

Table 4.3: The range of the standard deviation for  $\log_2(D)$

Dataset No.	$\mathbf{H}_1$	$\mathbf{H}_2$	$\mathbf{H}_3$	$\mathbf{H}_4$	Proposed
1	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$
2	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$
3	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$
4	$[10^{-3}, 10^{-2}]$	$[10^{-4}, 10^{-2}]$	$[10^{-4}, 10^{-2}]$	$[10^{-5}, 10^{-2}]$	$[10^{-5}, 10^{-2}]$
5	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-4}, 10^{-2}]$	$[10^{-4}, 10^{-2}]$	$\approx 10^{-3}$
6	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$	$[10^{-3}, 10^{-2}]$
7	$[10^{-7}, 10^{-1}]$	$[10^{-9}, 10^{-2}]$	$[0, 10^{-15}]$	$[0, 10^{-15}]$	$[0, 10^{-15}]$
8	$[10^{-5}, 10^{-2}]$	$[10^{-5}, 10^{-2}]$	$[10^{-9}, 10^{-4}]$	$[10^{-9}, 10^{-4}]$	$[10^{-8}, 10^{-4}]$

Figure 4.1-4.8 shows the average  $\log_2$  value of root-mean-squared residual  $D$  (i.e. reconstructive error  $D$ ) from these 20 runs of five different initialization methods with the increasing iteration number.  $D$  is the error which is equal to the  $\mathbf{F}_1$  shown in equation 3.2 and the lower  $D$  log value suggests the better performance for NMF. It is seen from the eight figures that all the  $D$  log values decrease fast in the early iterations of NMF and become stable at the end. The proposed initialization method  $\mathbf{H}_5$  always gets the lower  $D$  log values compared with the other four methods on the eight datasets

in the short-term. It shows that the  $D$  log values in the early iterations obtained by the five initialization methods always satisfy the inequality  $D(\mathbf{H}_1) > D(\mathbf{H}_2) > D(\mathbf{H}_3) = D(\mathbf{H}_4) > D(\mathbf{H}_5)$ . The method  $\mathbf{H}_1$  which initializes both the two NMF factors  $\mathbf{W}$  and  $\mathbf{H}$  randomly has the worst performance of NMF as we expect. The reason is that the random initialization has nothing to do with the initial factors  $\mathbf{W}$  and  $\mathbf{H}$  while the other four initialization methods already works for NMF algorithm with predefined factor  $\mathbf{W}$  (e.g. :  $\mathbf{H}_2$ ) or both the two factors (e.g. :  $\mathbf{H}_3, \mathbf{H}_4, \mathbf{H}_5$ ). So the  $D$  log value for random initialization is the largest. The methods  $\mathbf{H}_3, \mathbf{H}_4$  and  $\mathbf{H}_5$  adds the initiation process of the factor  $\mathbf{H}$ , so their  $D$  log values are smaller than  $\mathbf{H}_2$ . The methods  $\mathbf{H}_2, \mathbf{H}_3$  and  $\mathbf{H}_4$  start at the similar  $D$  log value, however,  $\mathbf{H}_3$  and  $\mathbf{H}_4$  need far fewer iterations to converge than  $\mathbf{H}_2$  on these eight datasets. From figure 4.1-4.8, we summarize that the proposed method  $\mathbf{H}_5$  achieves less error and faster convergence at the beginning of the NMF iterations compared with the other four standard initialization methods.

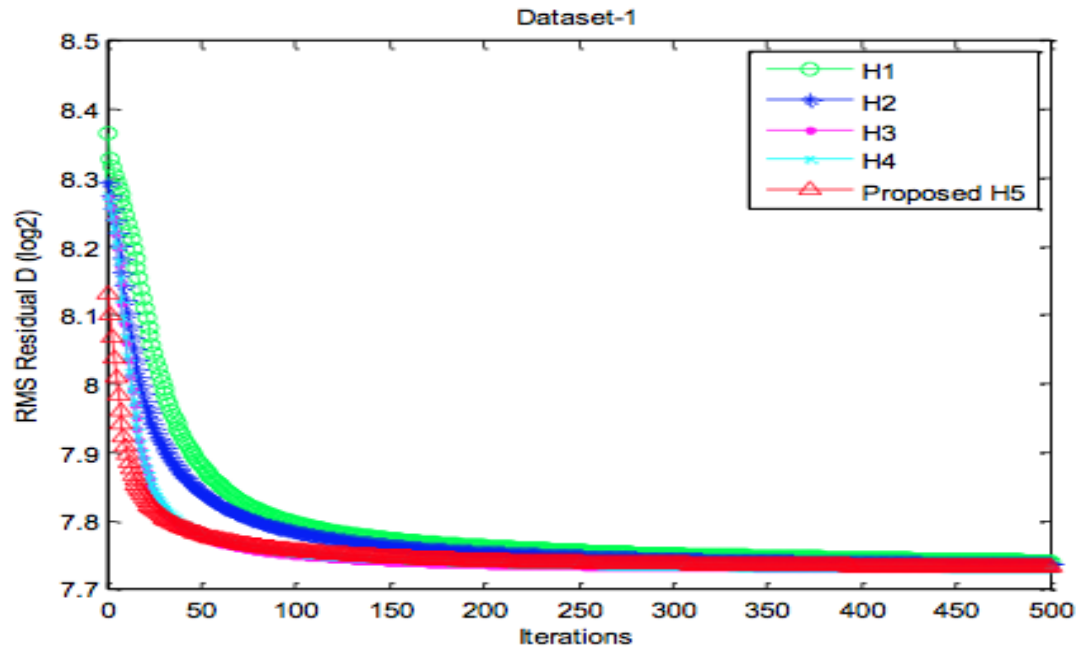


Figure 4.1: The average  $D$  log values of each of the five initialization methods (Dataset 1)

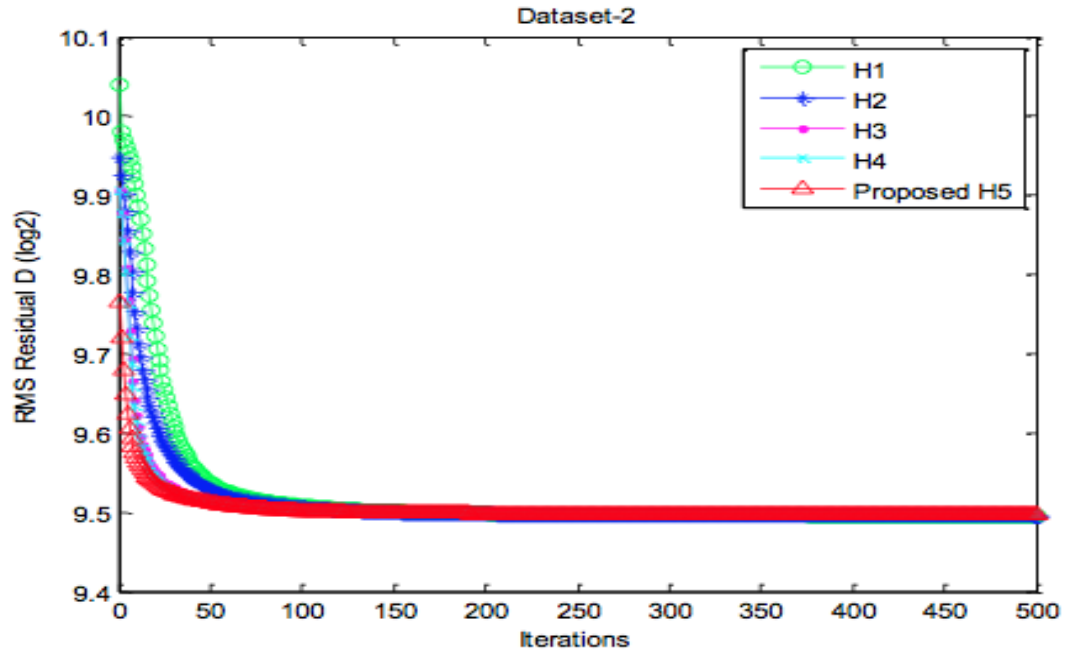


Figure 4.2: The average  $D$  log values of each of the five initialization methods (Dataset 2)

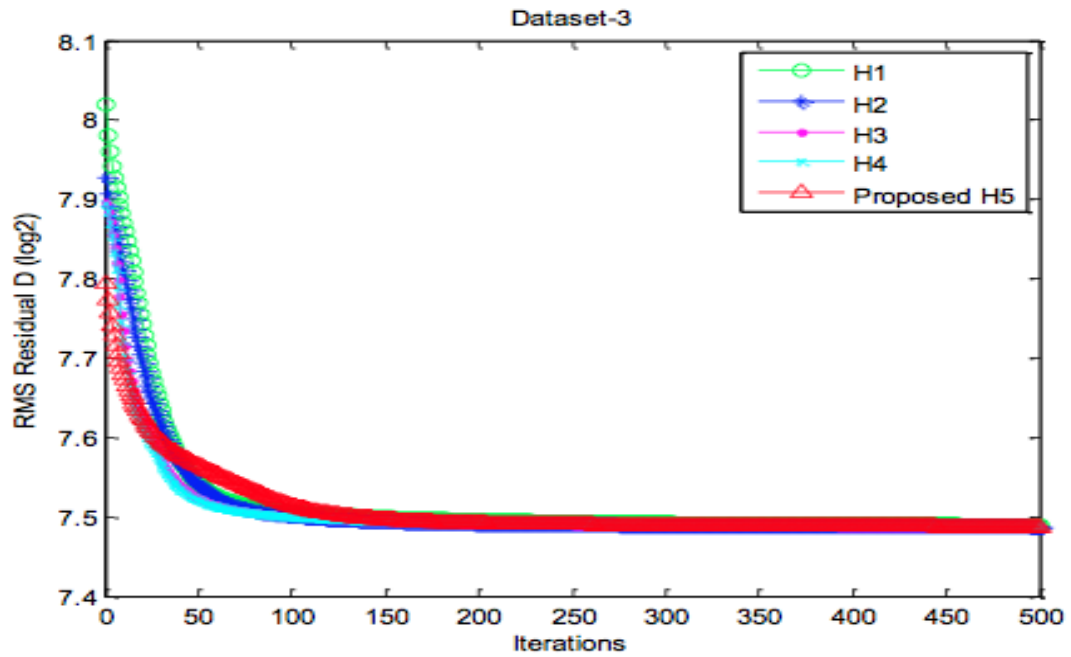


Figure 4.3: The average  $D$  log values of each of the five initialization methods (Dataset 3)

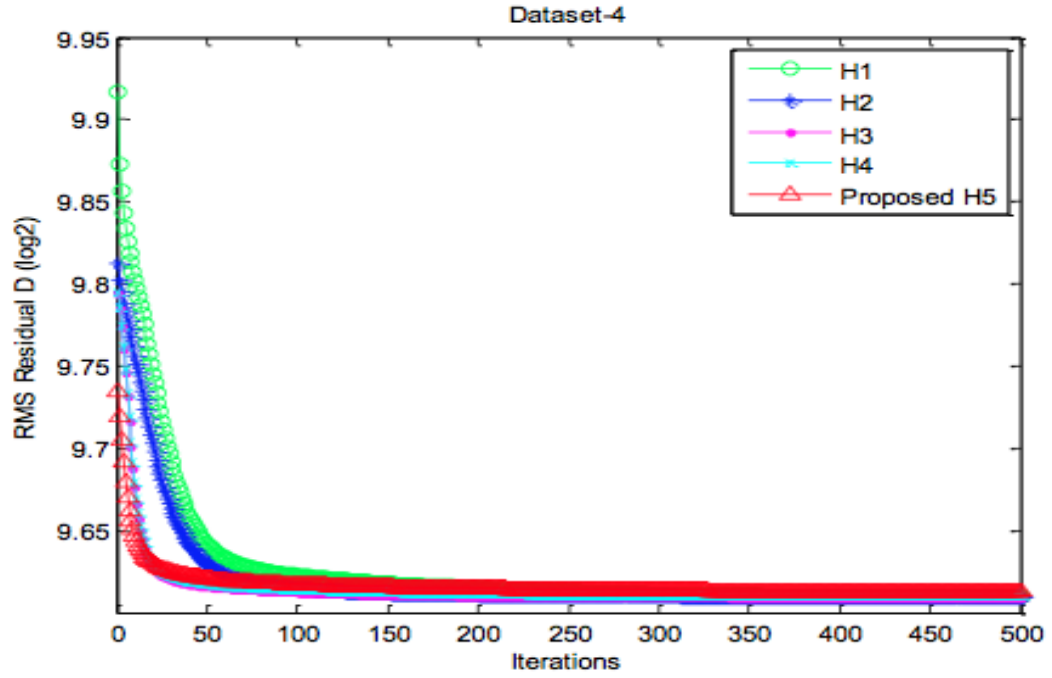


Figure 4.4: The average  $D$  log values of each of the five initialization methods (Dataset 4)

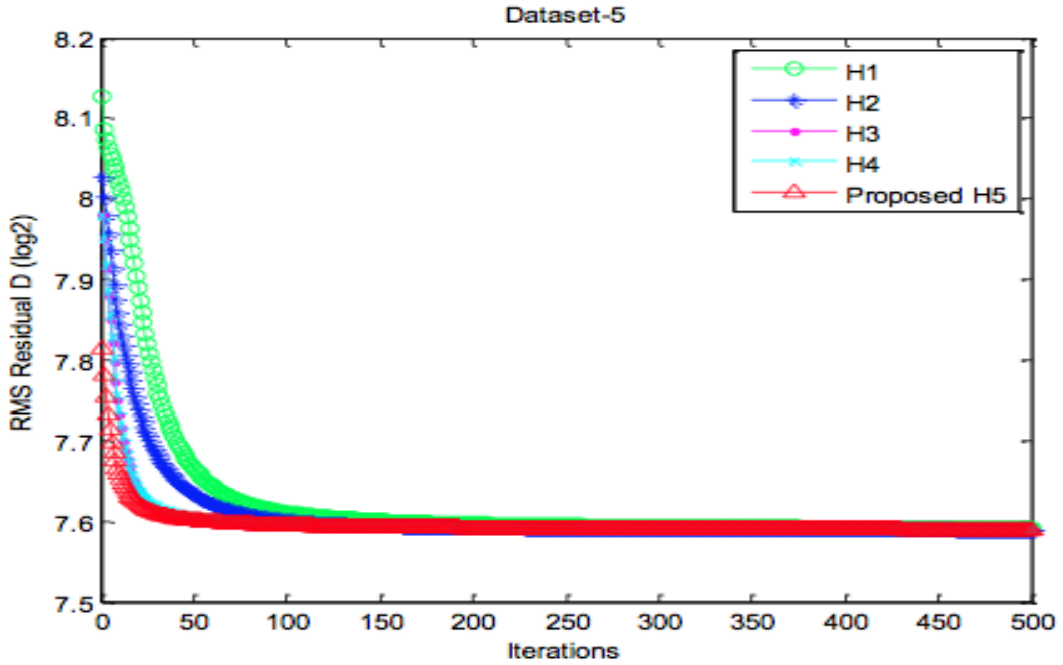


Figure 4.5: The average  $D$  log values of each of the five initialization methods (Dataset 5)



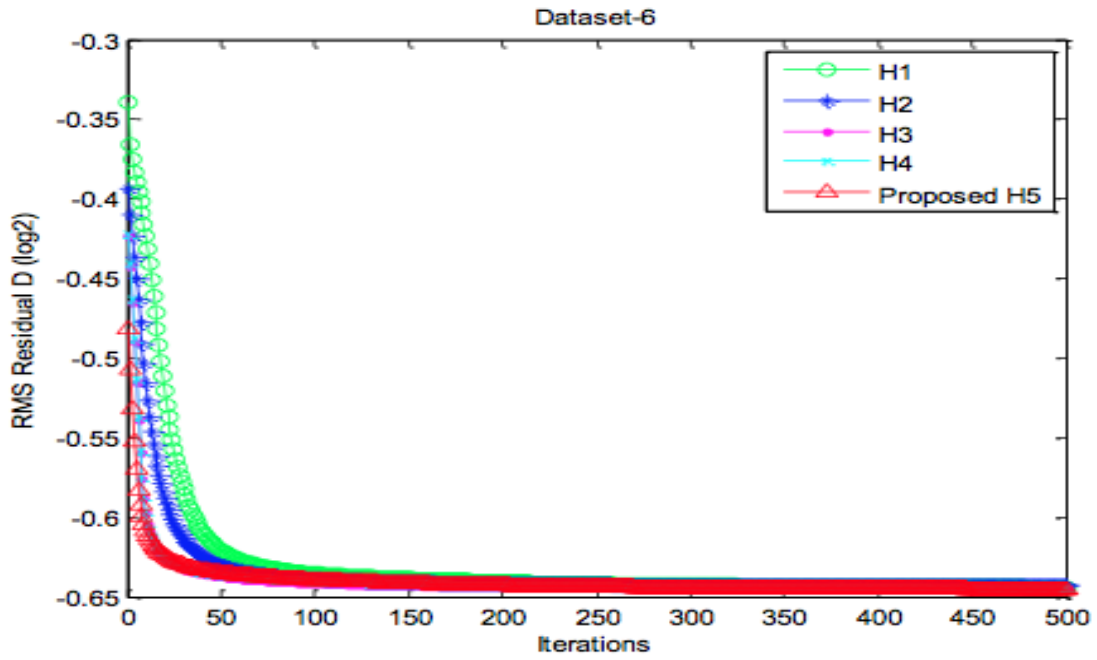


Figure 4.6: The average  $D$  log values of each of the five initialization methods (Dataset 6)

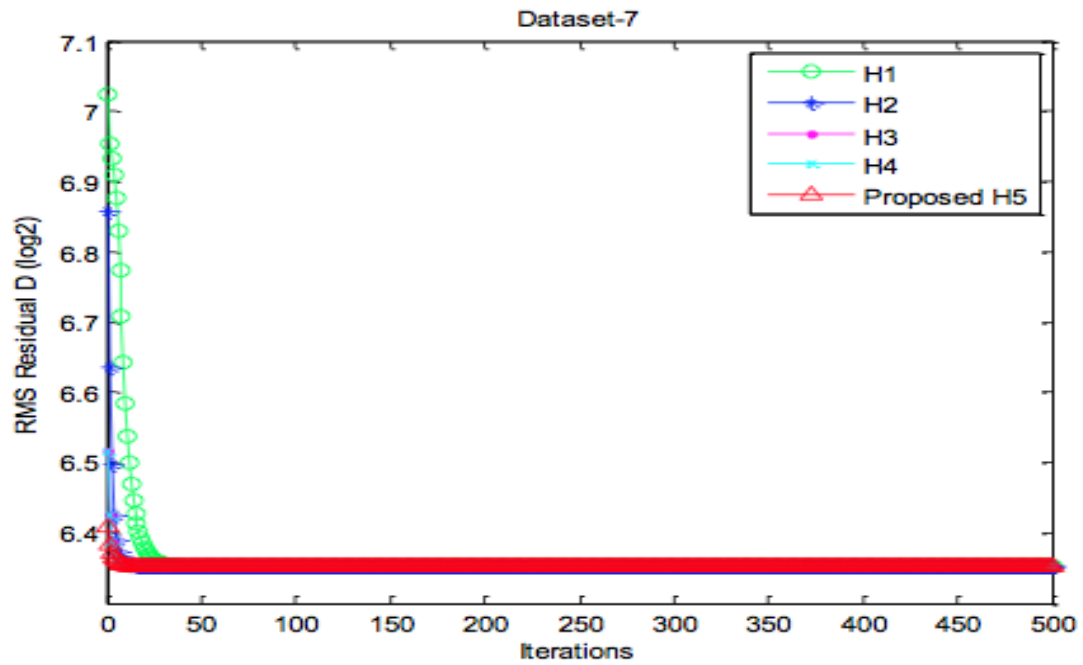


Figure 4.7: The average  $D$  log values of each of the five initialization methods (Dataset 7)

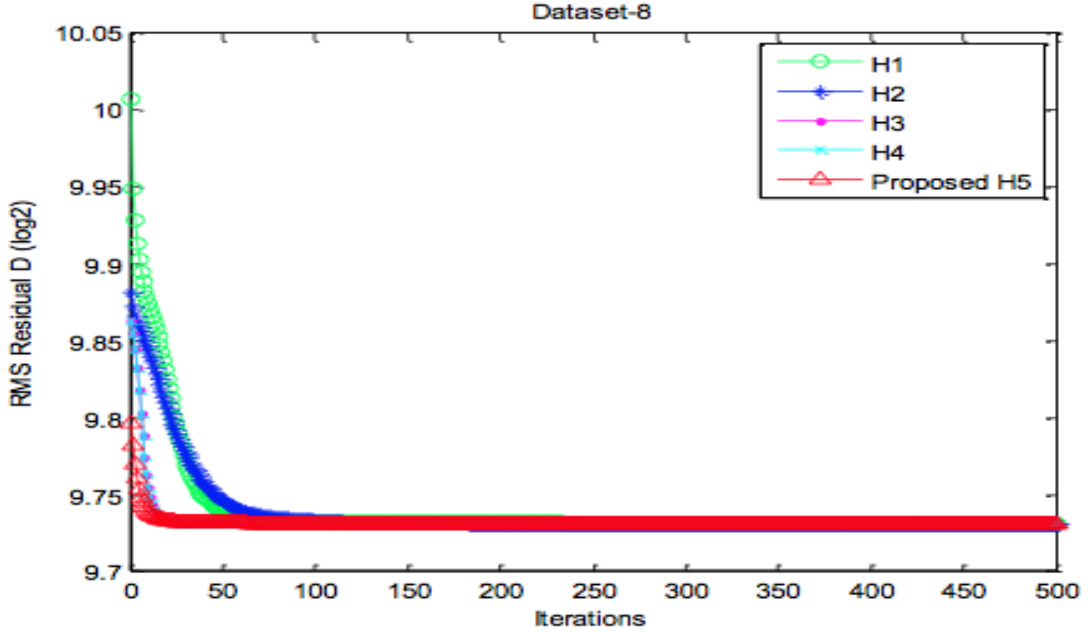


Figure 4.8: The average  $D$  log values of each of the five initialization methods (Dataset 8)

We have seen from the previous results that in the short-term, our proposed method (EIn-NMF) results in the better performance of NMF than the other four standard initialization methods. This property provides an idea to seek to benefit from the short-term behaviour for our proposed method (EIn-NMF). We have used the eight real datasets which already have the predefined rank  $k$  (the cluster number). However, the cluster number is unknown for most of the datasets in reality. For this reason, we analyze the RMS Residual  $D$  under the different dimensionality for each dataset. Here we calculate the NMF algorithm by using one iteration, because we expect to investigate the properties of the five methods at the start point of the NMF algorithm with the different dimensionality. The values of dimensionality for each dataset are set to  $\{5, 10, 15, \dots, [q], q\}$  where  $q$  is the row number of the dataset and  $[q]$  means the maximum integer which can be divided by 5 and smaller than  $q$ . In order to avoid the influence of the randomness, each dimensionality value under one initialization method was run 20 times and the number of iterations for each run of NMF was set to 1. The two figures below (figure 4.9 and 4.10) show the average RMS residual  $D$  values from these 20 runs of each of the five different initialization methods with the increasing number of dimensionality.

In these two figures, it can be easily seen that the four standard initialization methods ( $\mathbf{H}_1 - \mathbf{H}_4$ ) are almost stay at the higher  $D$  values while the proposed method  $\mathbf{H}_5$  keeps decreasing with the increasing dimensionality. For Dataset 1- 3, the last values of dimensionality are 60, 90 and 50 separately which are the same as their row numbers.

This means there is no change on the datasets after NMF algorithm. Only  $\mathbf{H}_1$  and the proposed  $\mathbf{H}_5$  can recognize this at the early iteration of NMF which have the zero  $D$  values at these dimensionality values. For Dataset 4-8 which the last dimensionality values are not the same as their row numbers, the proposed  $\mathbf{H}_5$  can still achieve the relative low  $D$  values while the four standard methods cannot. So we conclude the proposed  $\mathbf{H}_5$  always outperforms the other four standard initialization methods.

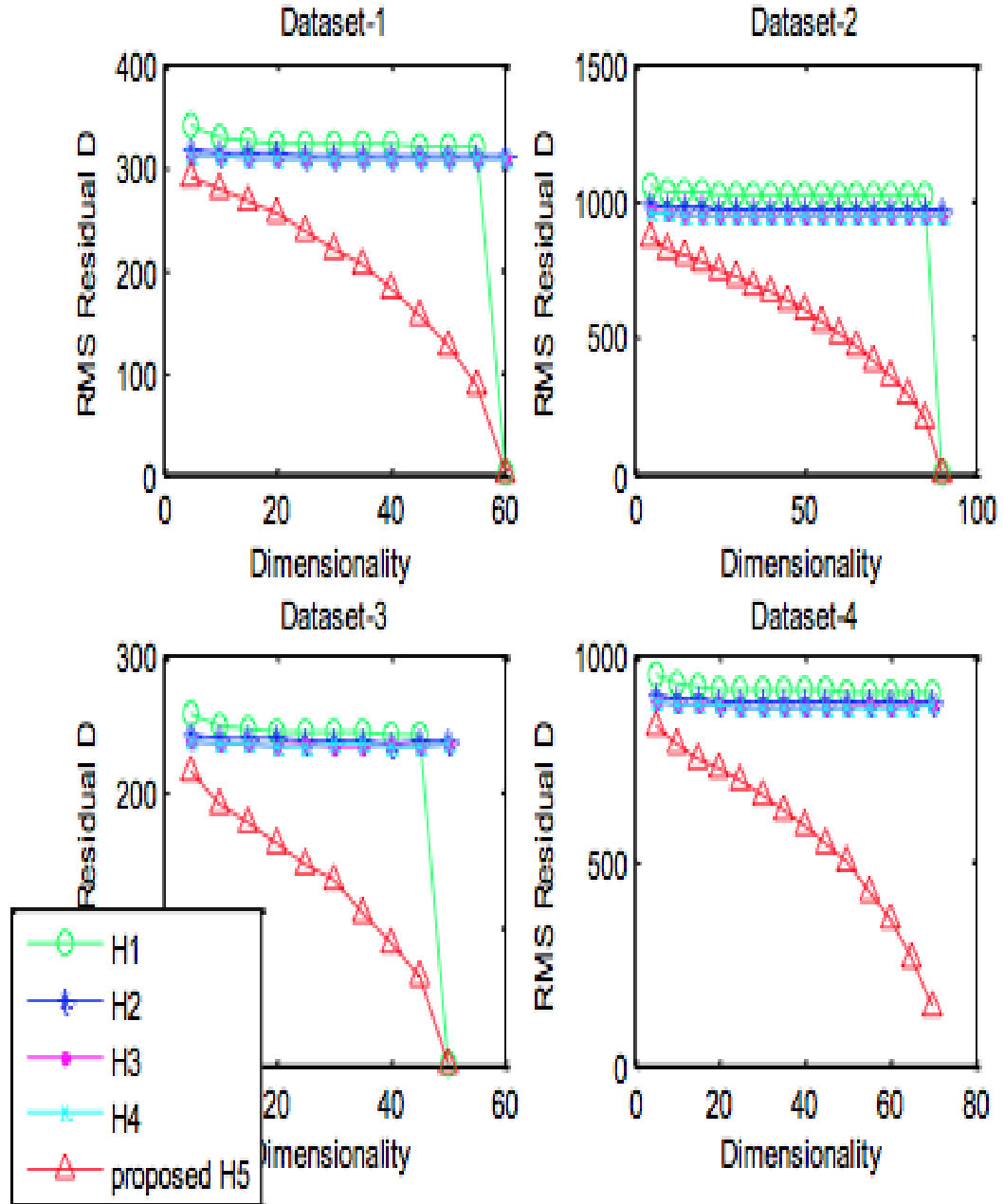


Figure 4.9: The average  $D$  values of each of the five initialization methods with the increasing dimensionality number on Dataset 1-4

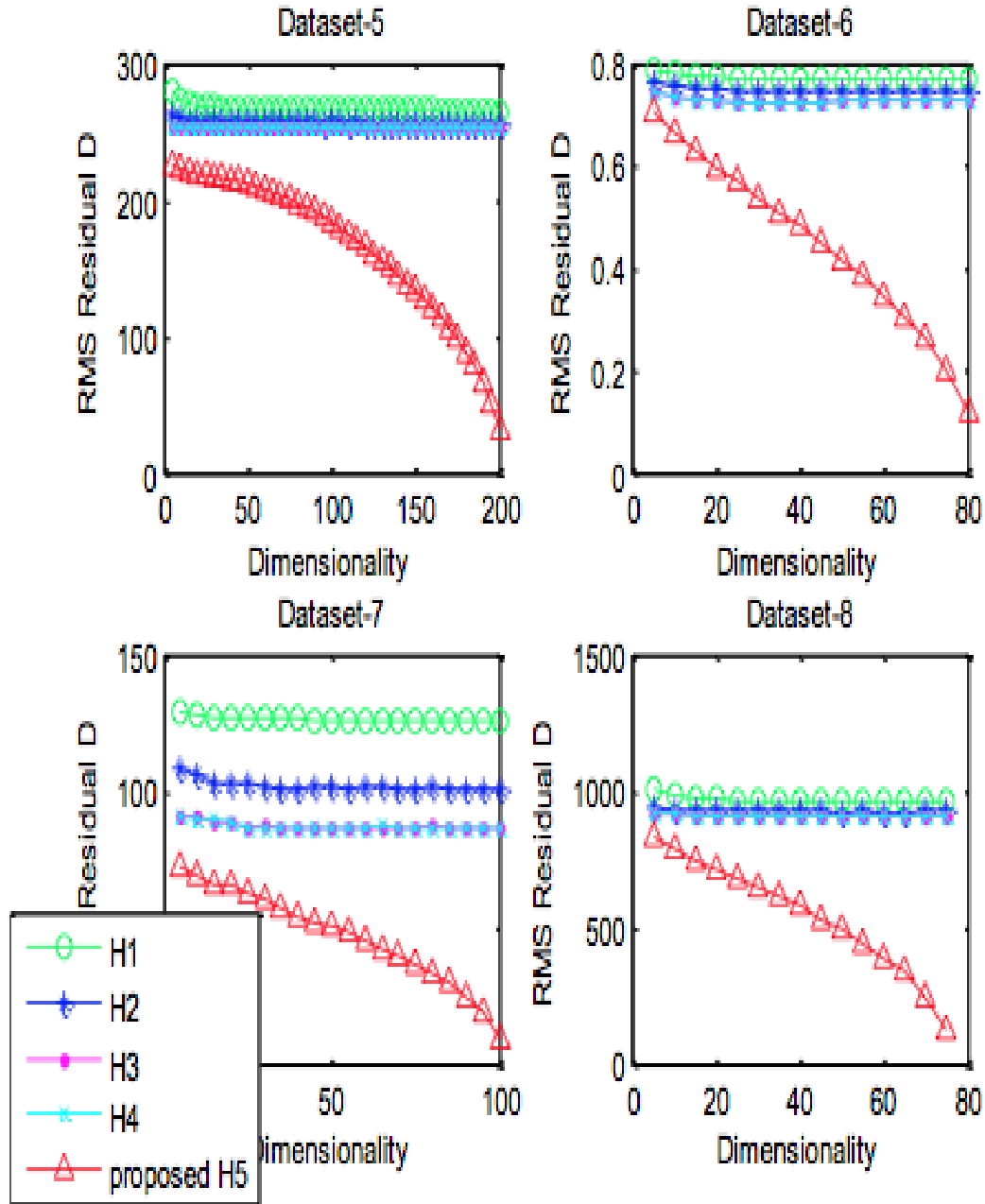


Figure 4.10: The average  $D$  values of each of the five initialization methods with the increasing dimensionality number on Dataset 5-8

### 4.2.3 Conclusion

In this section, we have proposed EIn-NMF, an initialization method of the factor  $\mathbf{H}$  for k-means based on NMF. Altogether, we also have compared our method with the other four different standard initialization methods—random initialization based NMF and three standard initialization methods for k-means based NMF. The experiments

were carried out on the eight real datasets from [61] and we assessed the NMF performance by the root-mean-squared residual  $D$ . The results demonstrate that the proposed initialization method, EIn-NMF, gets better performance of NMF compared with the other four standard initialization methods.

### 4.3 Dimensionality Reduction Based Initialization

Principal component analysis (PCA) and Independent component analysis (ICA) are two of the most popular dimensionality reduction methods used for visualizing high-throughput dataset in two or three dimensions. They keep the most information about dataset in the lower dimensional space so that the similarities within the dataset can be easily visualized (shown in section 3.3.3.1). Also Yao et al. has proposed independent principal component analysis (IPCA) which combines the advantages of both PCA and ICA [117]. It uses ICA as a de-noising process of the basic matrix produced by PCA to highlight the important structure of the dataset [117].

In this section, we explore the clustering performance of the NMF algorithm, with emphasis on the dimensionality reduction-based initialization problem. We propose an initialization method based on independent principal component analysis (IPCA) for NMF and results are compared with PCA-based initialization [122], ICA-based initialization and random initialization, using the RAND index [87]. The experiments were carried out on the several real datasets from UCI machine learning repository and the results showed that the IPCA-based initialization of NMF gets better clustering of the datasets compared with the other three methods.

The rest of the section is organized as follows. In Section 4.3.1 we describe the proposed IPCA-based initialization for NMF. Experimental results based on the four initialization methods (three standard and one new) are evaluated and analysed in Section 4.3.2. Finally, conclusion is drawn in Section 4.3.3.

#### 4.3.1 IPCA Based NMF

Yao et al. recently proposed an approach called IPCA which combines the advantage of both PCA and ICA [117]. ICA used in IPCA is a de-noising process of the basic matrix  $\mathbf{W}$  produced by PCA. Once the basic matrix  $\mathbf{W}$  is denoised, we expect it to be non-gaussian with no noise included. The research in [117] shows that IPCA outperforms PCA and ICA for solving some real datasets such as prostate cancer and yeast datasets. Also PCA and ICA have been successfully applied as the NMF initialization [122, 119]. So it is the way to thinking to use IPCA method as the initialization to furthermore de-noise the initial  $\mathbf{W}$  and improve the performance of NMF. The details of IPCA-based initialization method is described as follows.

1. Given the matrix  $\mathbf{X}$  as that of in NMF, and its transpose is set as  $\mathbf{A}$ . Apply

PCA on the matrix  $\mathbf{A}$  to generate the basic matrix  $\mathbf{W}$  (the same as PCA-based NMF).

2. Implement FastICA algorithm on this matrix  $\mathbf{W}$  and obtain the independent basic matrix (independent components)  $\mathbf{W}^*$ .
3. obtain the matrix  $\mathbf{H}^*$  which is calculated by

$$\mathbf{H}^* = (\mathbf{W}^*)^{-1}\mathbf{X} \quad (4.2)$$

4. We take the absolute value for all elements in  $\mathbf{W}^*$  and  $\mathbf{H}^*$ .
5. Apply NMF algorithm with these two initial values and obtain the cluster label of the given dataset from the final factor  $\mathbf{H}$ .

### 4.3.2 Results

The experiments were carried out by using the several datasets shown in Table 4.4. We applied four different initialization methods — random, PCA-based, ICA-based and IPCA-based initialization to improve the clustering performance of NMF. In order to avoid the influence of the randomness, each initialization method was run 20 times and the total number of iterations for each run of NMF was set to 500 in this experiment. The rank (dimensionality)  $k$  for each dataset is set to be the number of cluster of the corresponding dataset which is shown in Table 4.4.

Table 4.4: The properties of the datasets

Name	Rows	Columns	Cluster
Balance	625	4	3
Cancer-int	683	9	2
Credit	690	14	2
Dermatology	358	34	6
Diabetes	768	8	2
Iris	150	4	3
Thyroid	215	5	3
Wdbc	569	30	2
Wine	178	13	3

Table 4.5: The mean and standard deviation of initial RAND values of different initialization methods (iters=1, runs=20)

Name	Random	PCA-based	ICA-based	IPCA-based
Balance	53.0 ± 1.0	57.1 ± 0	52.7 ± 0.8	<b>62.9 ± 1.4</b>
Cancer-int	50.4 ± 0.5	58.8 ± 0	57.1 ± 5.9	<b>61.6 ± 0</b>
Credit	50.7 ± 0.7	52.8 ± 0	53.0 ± 2.5	<b>53.6 ± 0.8</b>
Dermatology	70.4 ± 0.2	75.8 ± 0	76.3 ± 3.8	<b>81.3 ± 2.4</b>
Diabetes	50.2 ± 0.3	<b>53.0 ± 0</b>	51.6 ± 1.6	<b>53.0 ± 0.1</b>
Iris	56.4 ± 0.7	73.5 ± 0	<b>78.4 ± 0</b>	78.3 ± 0
Thyroid	50.2 ± 1.5	58.7 ± 0	64.9 ± 7.2	<b>69.3 ± 6.2</b>
Wdbc	50.2 ± 0.2	<b>62.2 ± 0</b>	55.0 ± 6.6	61.6 ± 1.1
Wine	56.2 ± 0.8	72.5 ± 0	67.0 ± 3.0	<b>72.9 ± 0.9</b>

Table 4.6: The mean and standard deviation of final RAND values of different initialization methods (iters=500, runs=20)

Name	Random	PCA-based	ICA-based	IPCA-based
Balance	59.4 ± 4.4	57.1 ± 0	59.7 ± 4.3	<b>63.2 ± 0.2</b>
Cancer-int	63.5 ± 4.1	68.5 ± 0	61.5 ± 7.0	<b>69.1 ± 0.1</b>
Credit	51.6 ± 0.7	<b>53.0 ± 0</b>	52.3 ± 0.4	<b>53.0 ± 0</b>
Dermatology	83.5 ± 1.7	88.4 ± 0	87.4 ± 1.7	<b>89.1 ± 2.5</b>
Diabetes	53.1 ± 0.1	53.1 ± 0	<b>53.2 ± 0.2</b>	<b>53.2 ± 0.2</b>
Iris	77.4 ± 5.3	77.1 ± 0	80.4 ± 0	<b>80.6 ± 0</b>
Thyroid	74.5 ± 0.9	73.2 ± 0	73.4 ± 2.3	<b>75.7 ± 0.4</b>
Wdbc	67.6 ± 0.1	67.5 ± 0	67.5 ± 0.4	<b>67.7 ± 0.1</b>
Wine	64.7 ± 1.1	72.5 ± 0	68.5 ± 2.8	<b>72.6 ± 0.6</b>

Table 4.5 and 4.6 show the mean and the standard deviation of the initial and final RAND values from the 20 runs respectively. The bold values in the tables represent the best RAND value for each given dataset. We summarize that:

- IPCA-based initialization achieves the highest average RAND value in the short term and still remain the highest in the long term on most of the datasets.
- The standard deviation of the RAND values for PCA-based initialization is zero all times. This is because this initialization method computes the same initial values of  $\mathbf{W}$  and  $\mathbf{H}$  each time.
- The random initialization has nothing to do with the initial values of  $\mathbf{W}$  and  $\mathbf{H}$  while the PCA-based, ICA-based and IPCA-based initialization already works for clustering with predefined values of  $\mathbf{W}$  and  $\mathbf{H}$ , so the initial average RAND value of random initialization is the smallest on most of the datasets.
- IPCA-based initialization adds the advantage of both PCA and ICA, so its RAND values are larger than PCA-based and ICA-based initialization.
- Sometimes the random initialization achieves the higher final RAND values than the PCA-based initialization. This is because NMF algorithm with PCA-based

initialization using Euclidean distance measure cannot pull the factorization out of local minima in these datasets. However, IPCA-based initialization can perform well which has the even higher RAND value than random initialization.

Figure 4.11 to 4.19 show the initial and final RAND values at the 1<sup>st</sup> iteration and 500<sup>th</sup> iteration from the 20 runs respectively. Also the whole RAND values with the increasing iterations are recorded.

- We can see that the most of the final RAND values of the random initialization in credit, dermatology and wine are lower than the other methods.
- In balance and iris datasets, although the PCA-based initialization enhances the initial values of  $\mathbf{W}$  and  $\mathbf{H}$ , it still gets the lower RAND values than the random initialization after number of iterations. However, the IPCA-based initialization can solve this problem which has the higher clustering performance than the random initialization all the time.
- In wdbc dataset, the IPCA-based initialization keeps the highest RAND values at the head start and maintains this advantage at the end. In this case, IPCA-based initialization can be used in the short term with the less computational complexity.



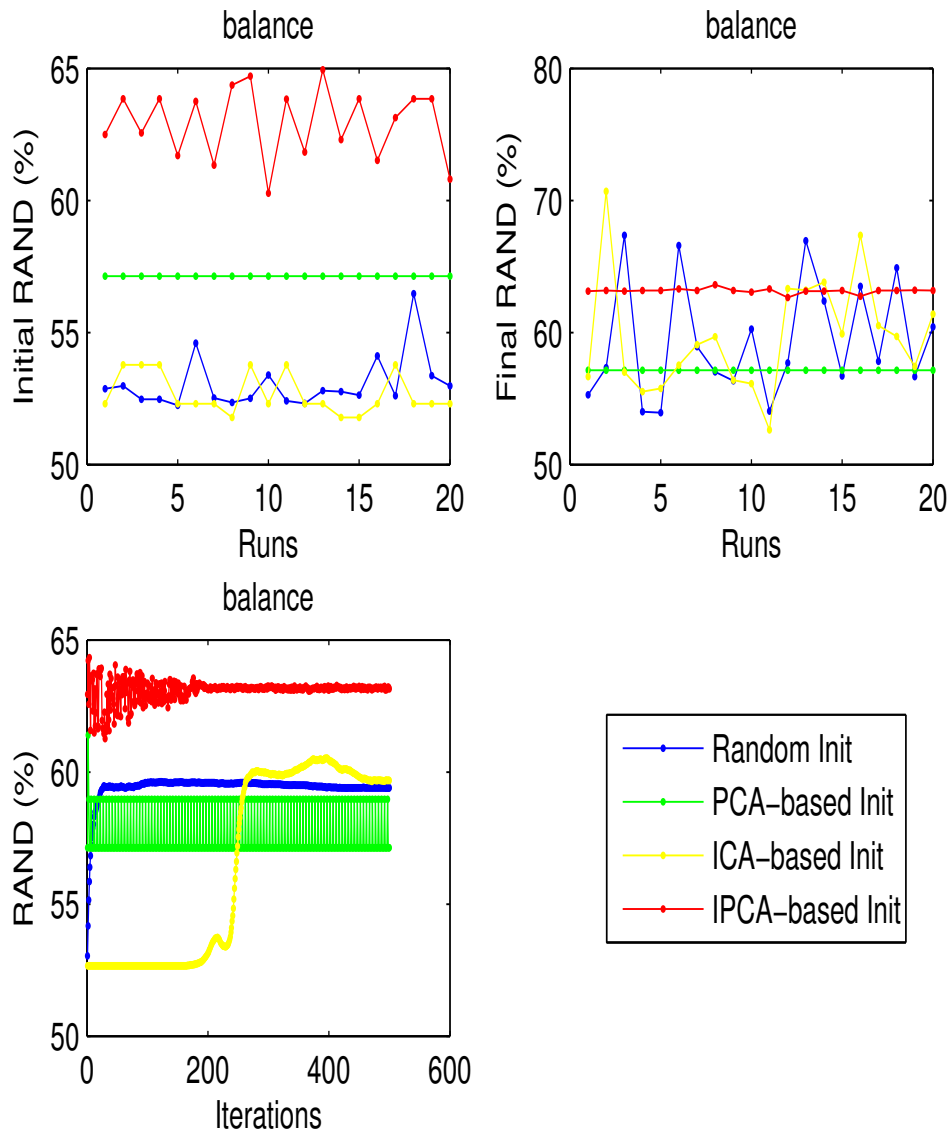


Figure 4.11: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for balance dataset

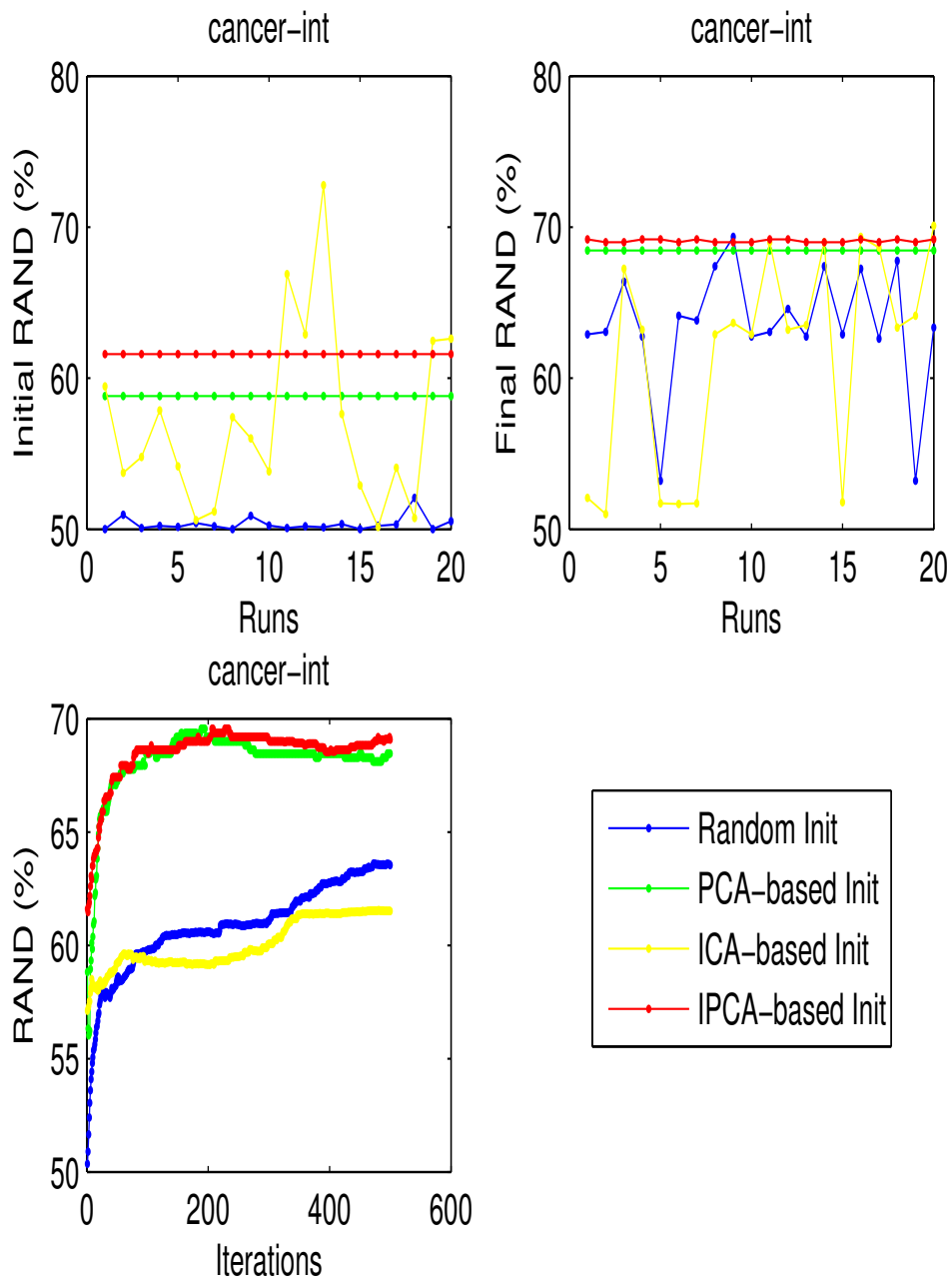


Figure 4.12: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for cancer-int dataset

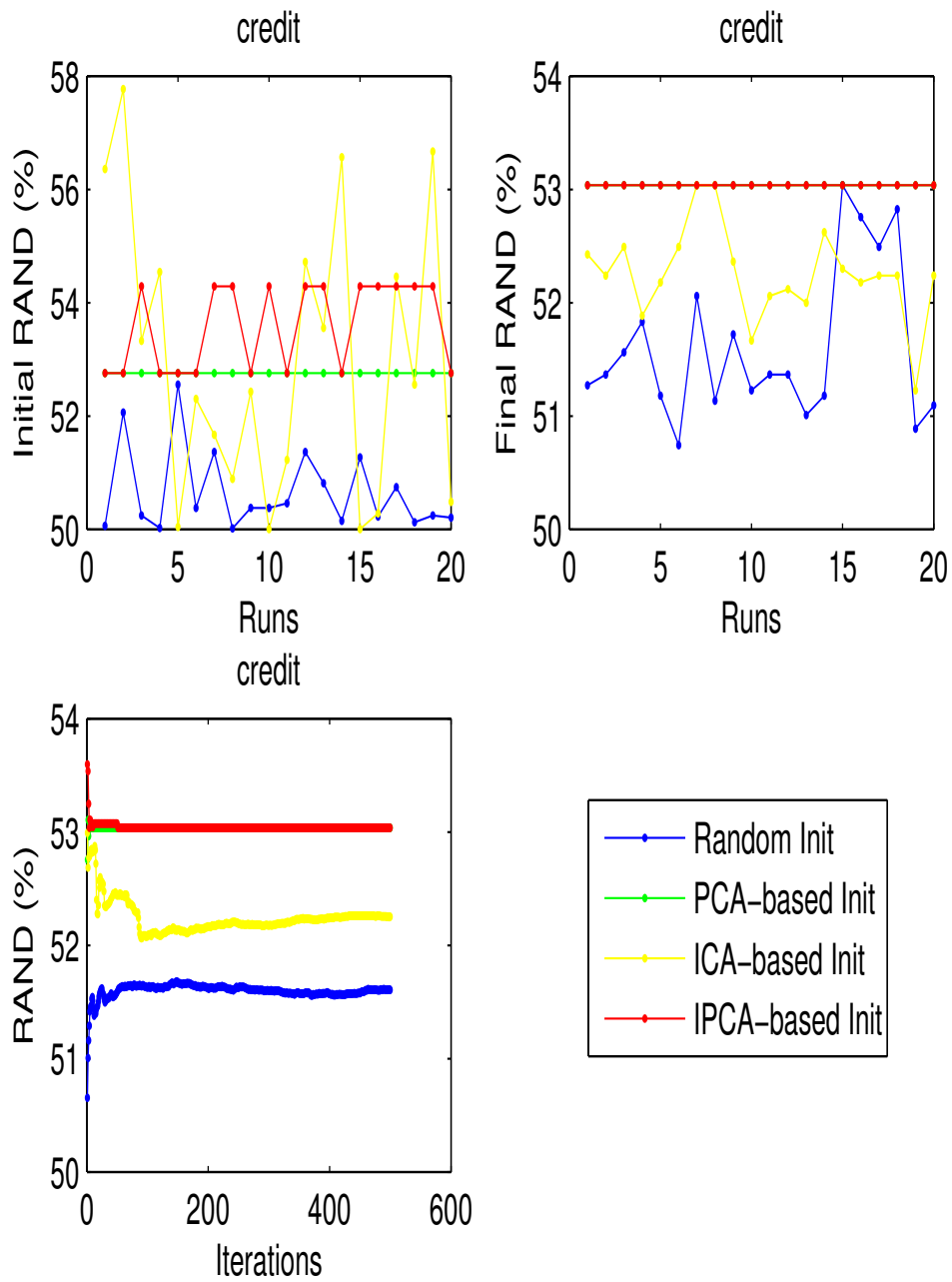


Figure 4.13: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for credit dataset

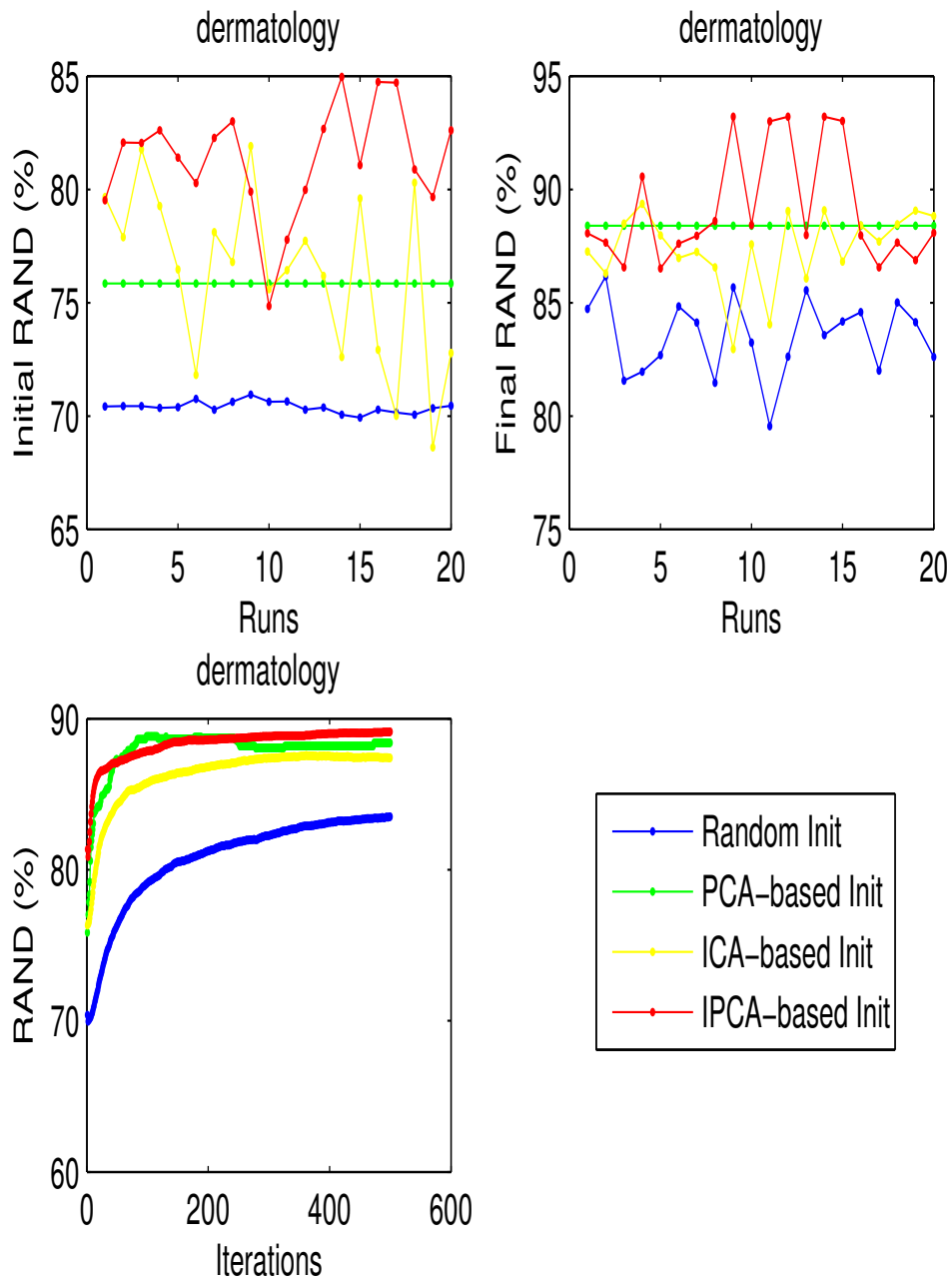


Figure 4.14: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for dermatology dataset

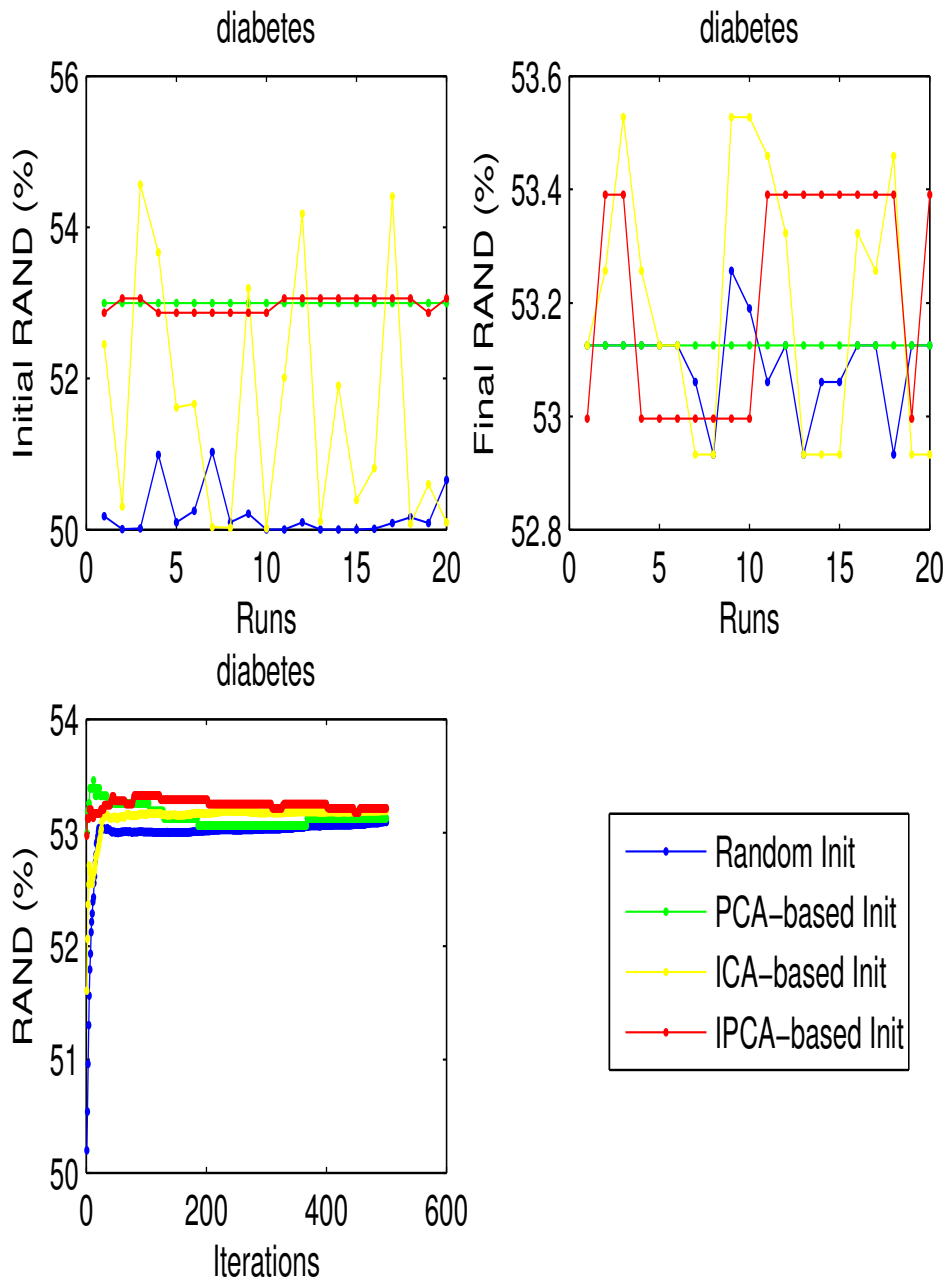


Figure 4.15: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for diabetes dataset

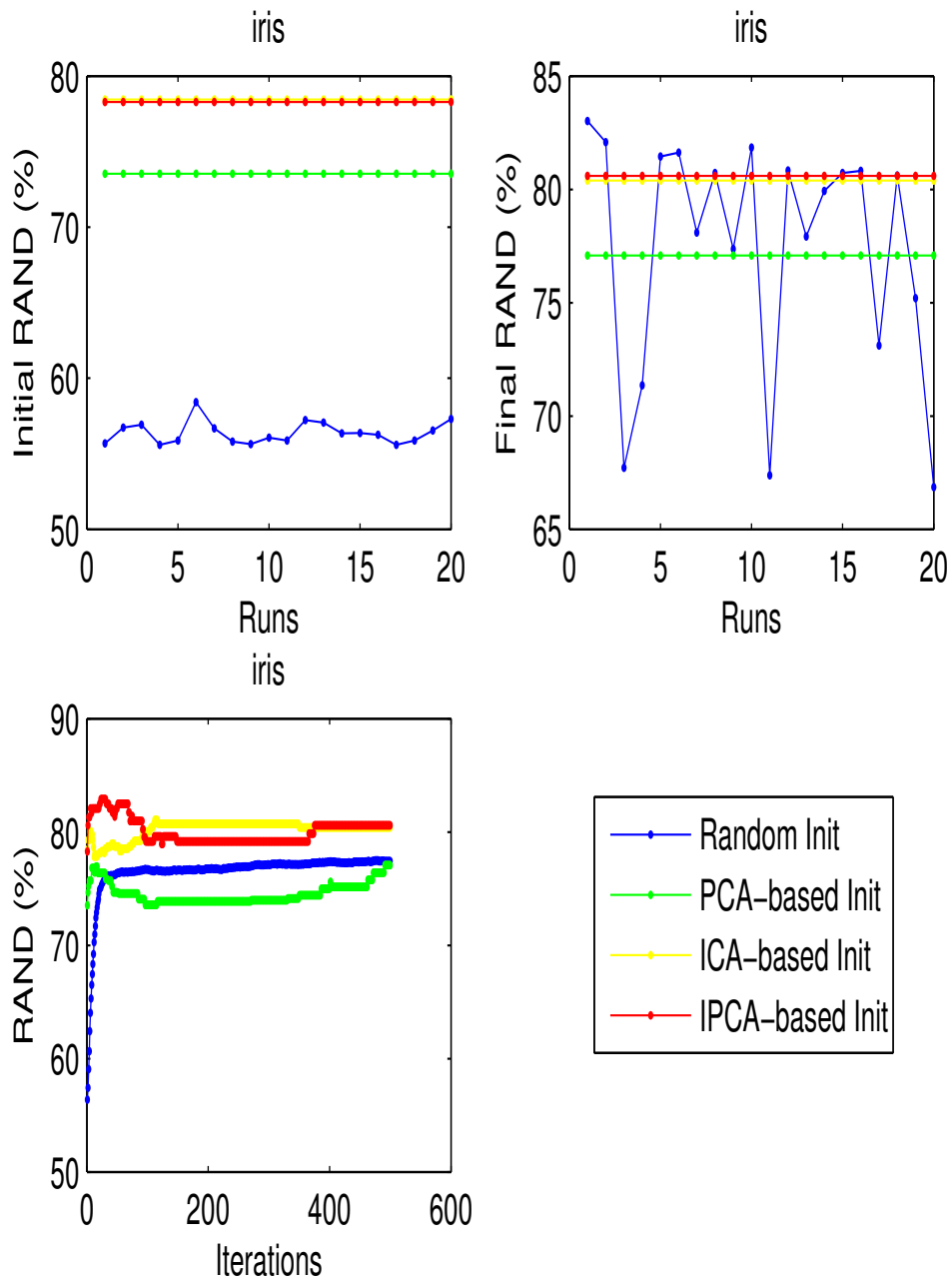


Figure 4.16: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for iris dataset

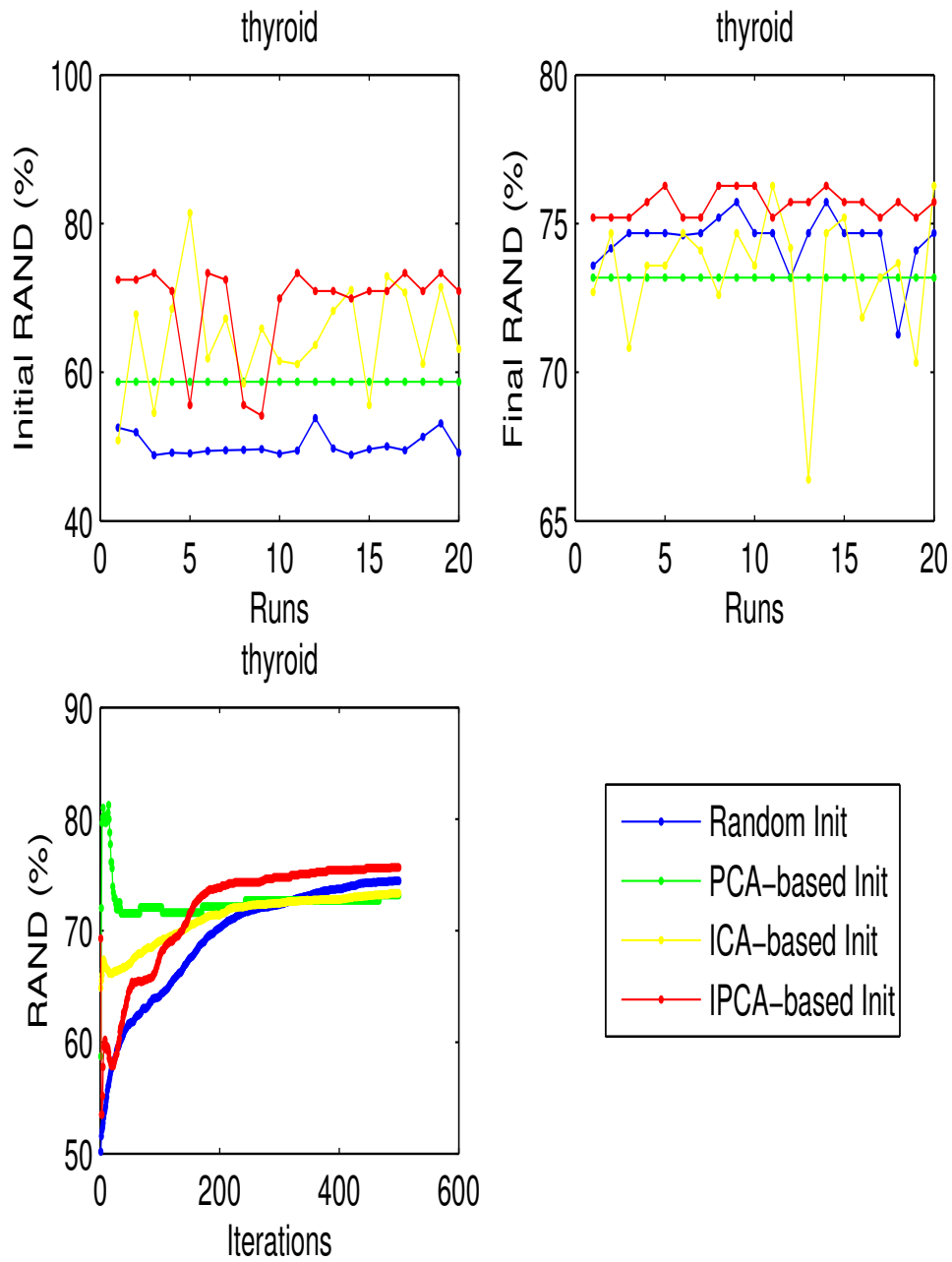


Figure 4.17: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for thyroid dataset

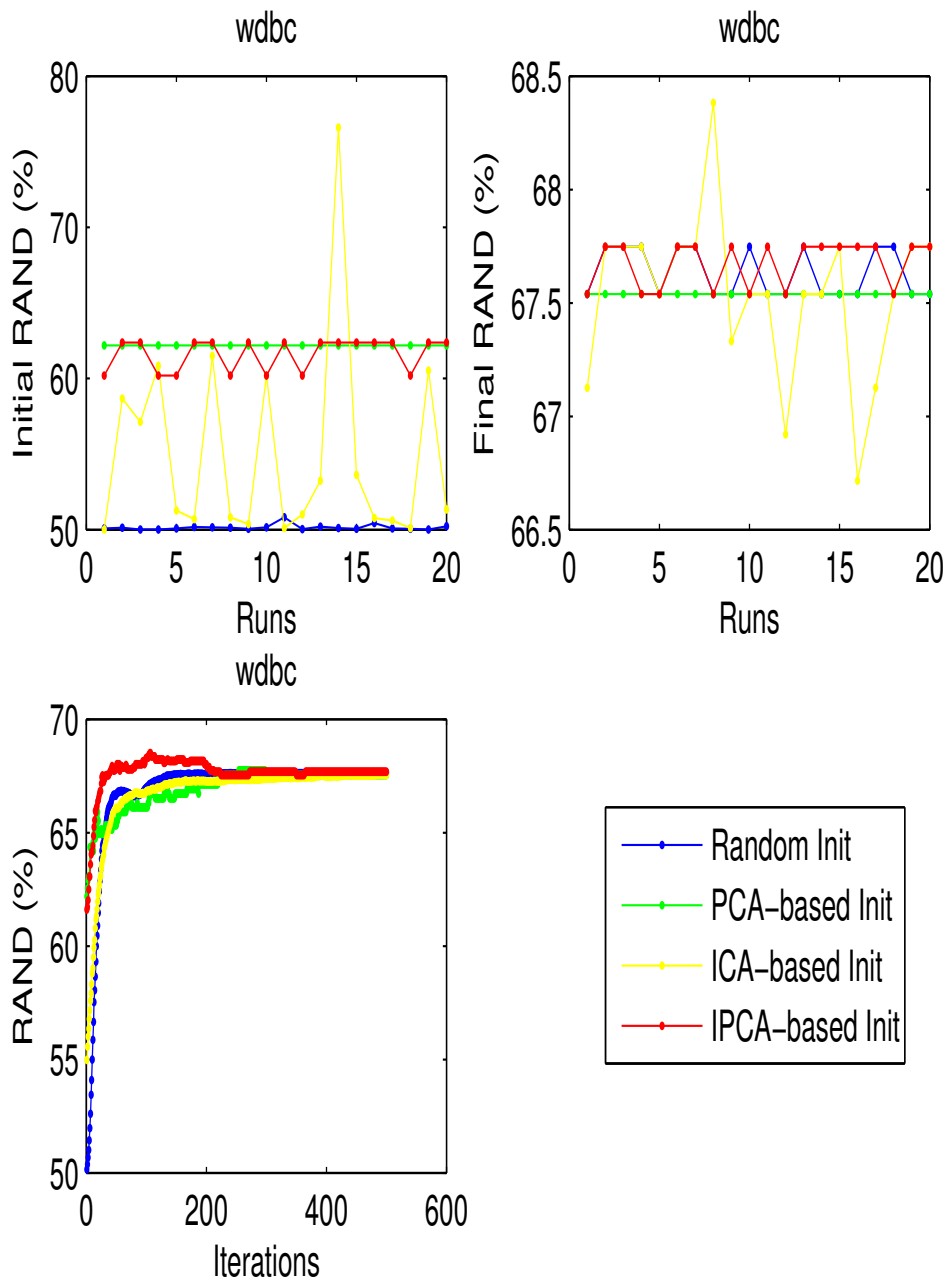


Figure 4.18: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for wdbc dataset



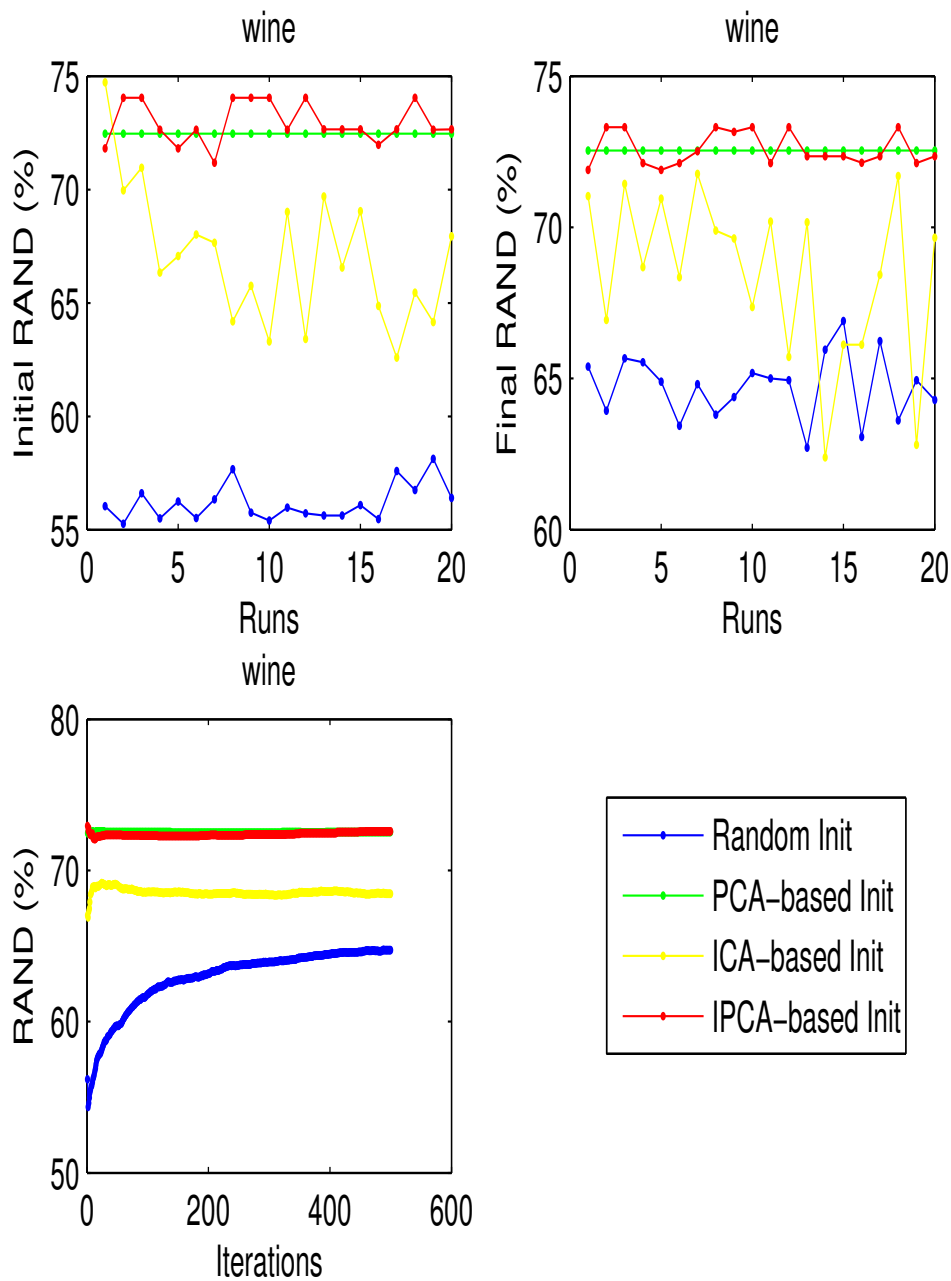


Figure 4.19: The clustering performance of random, PCA-based, ICA-based and IPCA-based initialization for wine dataset

### 4.3.3 Conclusion

Researchers often use random initializations when utilizing NMF. To improve the performance of NMF, we have proposed an initialization method based on IPCA for NMF in this section. Altogether, we also have explored the NMF algorithm with the three different initialization methods. The initialization methods are based on random, PCA,

and ICA. The experiments were carried out on several real datasets from UCI machine learning repository and we assessed the clustering performance using the RAND index [87]. From the experimental results, we see that the IPCA-based NMF can achieve faster convergence in the datasets. Also we conclude that the proposed IPCA-based initialization of NMF gets better clustering of the datasets compared with the random, PCA-based and ICA-based initialization. Here we only compared the four initialization methods (three standard and one new) together. As there are other good initialization methods in the literature, comparing these initialization methods would be considered in the future work.

## Chapter 5

# Proposed NMF Updating Strategy for Data Clustering

This chapter propose the new evolutionary optimization strategy for NMF driven by three proposed evolving rules in the solution space, saying NMF rule, firefly rule and survival of the fittest rule. This proposed update strategy facilitates the clustering problem by using the system objective functions that make use of the clustering quality measurement. Work of this chapter is prepared as the following paper.

Liyun Gong, Tingting Mu, Meng Wang, Hengchang Liu and John Y. Goulermas, A robust nonnegative matrix factorization strategy with adaptive quality control of data clusters, *Pattern Recognition*, 2014. (IF=2.632, under review)

### 5.1 Introduction

Nonnegative matrix factorization (NMF) enables approximation of data vectors by linear combinations of a small set of nonnegative basis vectors weighted by nonnegative coefficients. A common approach of clustering analysis is to group the data vectors based on the resulting coefficients. However, these coefficients are not necessarily to always present an improved cluster structure. To achieve better use of NMF for data clustering, this chapter proposes a novel evolutionary strategy to improve the iterative updating procedure of NMF, aiming at producing higher-quality coefficients that offer more accurate and sharpened cluster structure. Hybridization of multiple initialization approaches is enabled to improve the robustness of the solution. Three evolving rules are designed so that the cluster quality and reconstruction error are simultaneously improved during the updates. Any measure of clustering performance, e.g., either an internal one totally replying on the original data or an external one with the assist of extra information on ground truth partition, is allowed to drive the evolving procedure. Effectiveness of the proposed method is demonstrated via careful experimental design and thorough comparatively analysis using nine benchmark datasets.

The rest of the chapter is organized as follows: Section 5.2, the proposed method is

described. Experimental results and analysis are provided in Section 5.3.

## 5.2 Proposed Method

In this work, we propose an evolutionary strategy to improve the iterative updating procedure of NMF, which is named as ENMF and is aiming at producing higher-quality basis and encoding coefficient matrices  $\mathbf{W}$  and  $\mathbf{H}$  to facilitate the data clustering task. The algorithm starts from multiple pairs of initialization matrices for the basis and encoding coefficient matrices, which form an initial candidate set denoted as  $S_0 = \{(\mathbf{W}_0^i, \mathbf{H}_0^i)\}_{i=1}^m$  where  $\{\mathbf{W}_0^i\}_{i=1}^m$  and  $\{\mathbf{H}_0^i\}_{i=1}^m$  are referred as the seed matrices. The algorithm then evolves, creating an updated candidate set at each iteration, denoted as  $S_t = \{(\mathbf{W}_t^i, \mathbf{H}_t^i)\}_{i=1}^{m_t}$  for the  $t$ th iteration with  $m_t$  denoting the new candidate number. In the end, the optimal encoding coefficient matrix and its corresponding basis matrix are selected from the finally evolved candidate set based on a score function formulated to suit the data clustering task.

### 5.2.1 Seed Matrix Generation

To take advantage of the state-of-the-art NMF initialization strategies and to achieve local improvement of the optimal solution, multiple NMF initialization approaches are utilized to construct the initial candidate set, which contains various seed matrices of the basis and encoding coefficient ones:

- The CI approach is first conducted via performing k-means clustering [36]. The resulting binary cluster membership matrix  $\mathbf{M}$  is used as  $\mathbf{H}_0^1$ , and the resulting clustering centroid matrix  $\mathbf{C}$  as  $\mathbf{W}_0^1$ .
- Similar CI approach is conducted again but via fuzzy c-means (FCM) clustering [10]. The obtained cluster membership and centroid matrices  $\mathbf{M}$  and  $\mathbf{C}$  are used as  $\mathbf{H}_0^2$  and  $\mathbf{W}_0^2$ , respectively. In addition, one more candidate is generated by setting the  $n \times k$  member degree matrix  $\mathbf{U} = [u_{ij}]$  of FCM as  $\mathbf{H}_0^3$  and the same centroid matrix  $\mathbf{C}$  as  $\mathbf{W}_0^3$ . Here, the degree value  $u_{ij}$  represents the confidence value the  $i$ th data point belonging to the  $j$ th cluster and satisfies the conditions of  $0 \leq u_{ij} \leq 1$  and  $\sum_{j=1}^k u_{ij} = 1$ .
- The RI and RAI approaches are used to generate the two candidates of  $(\mathbf{W}_0^4, \mathbf{H}_0^4)$  and  $(\mathbf{W}_0^5, \mathbf{H}_0^5)$ .

It is worth to note that the proposed NMF updating algorithm is a general method. The users can freely choose any type and any number of initial candidates to suit their needs apart from the above ones.

## 5.2.2 Evolving Strategy

In each iteration, three new subsets of candidates  $S_{t+1}^{(M)}$ ,  $S_{t+1}^{(F)}$  and  $S_{t+1}^{(S)}$  are generated from the previous set  $S_t$ , according to three types of evolving rules proposed, namely the NMF rule, firefly rule and the survival of the fittest rule, respectively. The three subsets together constitute the updated set  $S_{t+1} = S_{t+1}^{(M)} \cup S_{t+1}^{(F)} \cup S_{t+1}^{(S)}$  at the  $(t+1)$ th iteration. In the following, we explain these rules in detail.

### 5.2.2.1 Multiplicative rule

This rule is constructed to take advantage of the classical multiplicative update rules for NMF approximation. It generates the new candidate subset by

$$S_1^{(M)} = \Phi_1(S_0, \mathbf{X}), \quad (5.1)$$

for the first iteration and

$$S_{t+1}^{(M)} = \Phi_1(S_t^{(M)}, \mathbf{X}), \quad (5.2)$$

for the  $(t+1)$ th iteration ( $t \geq 1$ ). The operation  $S' = \Phi_1(S, \mathbf{X})$  takes one set of matrix pairs  $S = \{(\mathbf{W}_i, \mathbf{H}_i)\}_{i=1}^m$  and one  $d \times n$  data matrix  $\mathbf{X}$  as the input, where each matrix pair includes one  $d \times k$  matrix  $\mathbf{W}_i$  and one  $n \times k$  matrix  $\mathbf{H}_i$ , and outputs a set of matrix pairs denoted as  $S' = \{(\mathbf{W}'_i, \mathbf{H}'_i)\}_{i=1}^m$ . It is formulated based on Eqs. 3.3 such as

$$\mathbf{H}'_i = \mathbf{H}_i \circ (\mathbf{X}^T \mathbf{W}_i) \oslash (\mathbf{H}_i \mathbf{W}_i^T \mathbf{W}_i), \quad (5.3)$$

$$\mathbf{W}'_i = \mathbf{W}_i \circ (\mathbf{X} \mathbf{H}_i) \oslash (\mathbf{W}_i \mathbf{H}_i^T \mathbf{H}_i). \quad (5.4)$$

where  $\circ$  denotes the Hadamard product and  $\oslash$  the Hadamard division of two matrices of the same size,  $\mathbf{W}_i$  and  $\mathbf{H}_i$  denote the computed basis and encoding coefficient matrices for the  $i^{\text{th}}$  candidate. This rule enables the inclusion of multiple NMF solutions obtained by the multiplicative update rules, which are driven by the same reconstruction error minimization but initialized through different ways, e.g., the random and clustering-based ones, to the final evolved candidate set.

### 5.2.2.2 Firefly Rule

This rule encourages the generation of new candidate matrix pairs that may contain higher quality of encoding coefficient matrix than those obtained by the previous multiplicative rule, in order to facilitate the data clustering task more effectively.

In the first iteration, the firefly rule operates on the candidate subset  $S_1^{(M)}$  generated by the multiplicative rule, and further creates another candidate subset by

$$S_1^{(F)} = \Phi_2(S_1^{(M)}, \mathbf{H}_1^*). \quad (5.5)$$

The operation  $S' = \Phi_2(S, \mathbf{A})$  takes a set  $S = \{(\mathbf{W}_i, \mathbf{H}_i)\}_{i=1}^m$  and an  $n \times k$  matrix  $\mathbf{A}$  as input, while outputs a new set  $S' = \{(\mathbf{W}'_i, \mathbf{H}'_i)\}_{i=1}^m$ . The corresponding relationship

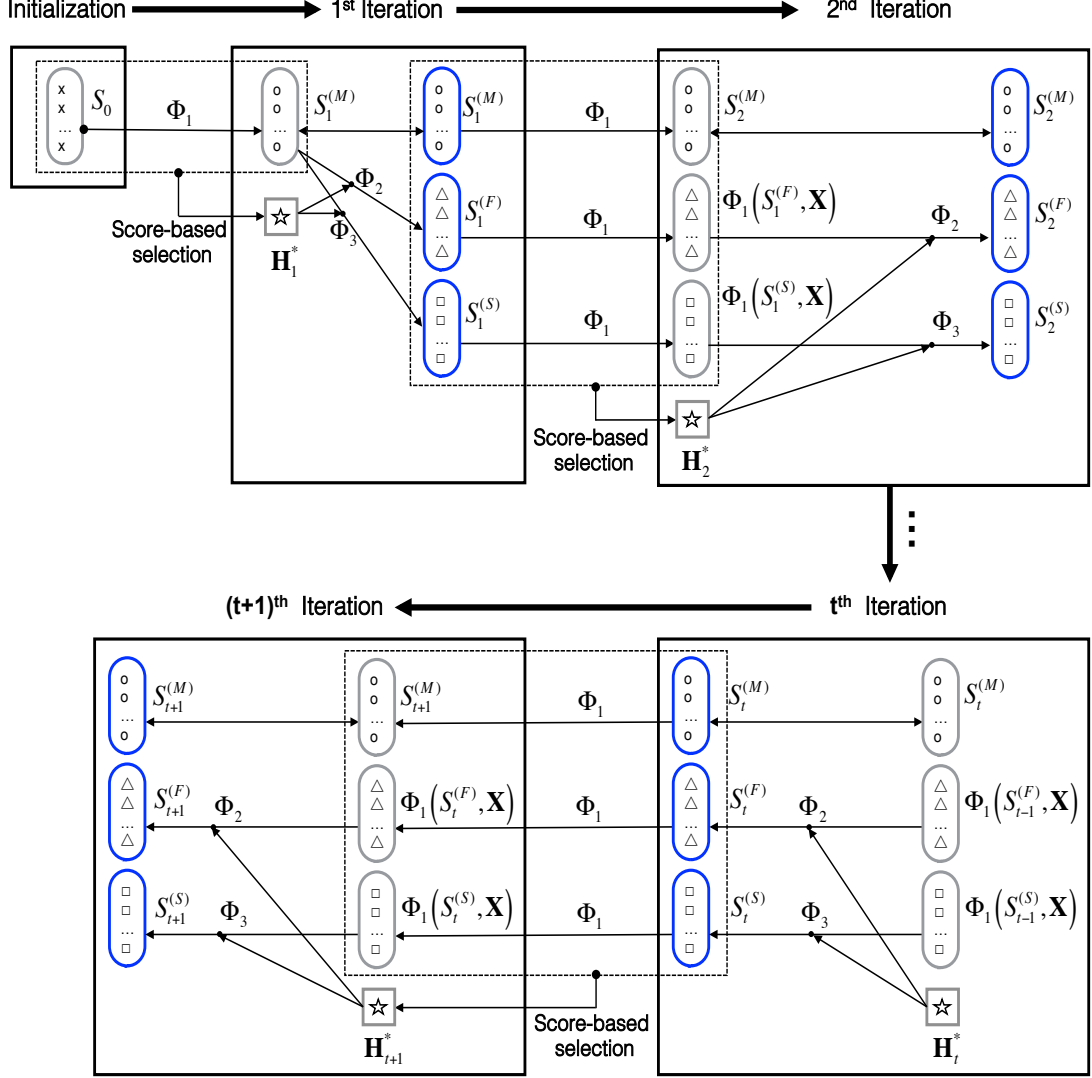


Figure 5.1: Data flow of the proposed ENMF. The circle, triangle and rectangle symbols represent candidates derived during the generation of the  $S_i^{(M)}$ ,  $S_i^{(F)}$  and  $S_i^{(S)}$  subsets, respectively.

between its input and output is defined by

$$\mathbf{H}'_i = \mathbf{H}_i + \beta \exp^{-\gamma \|\mathbf{A} - \mathbf{H}_i\|_F^2} (\mathbf{A} - \mathbf{H}_i), \quad (5.6)$$

$$\mathbf{W}'_i = \begin{cases} \tilde{\mathbf{W}}_i, & \text{if } \|\mathbf{X} - \tilde{\mathbf{W}}_i \mathbf{H}'_i{}^T\|_F^2 < \|\mathbf{X} - \mathbf{W}_i \mathbf{H}'_i{}^T\|_F^2 \\ \mathbf{W}_i, & \text{otherwise.} \end{cases} \quad (5.7)$$

In the above equations,  $0 < \beta \leq 1$  and  $\gamma > 0$  are parameters set by the user, and the matrix  $\tilde{\mathbf{W}}_i$  is computed by

$$\tilde{\mathbf{W}}_i = \max\left(0, \mathbf{X} \left(\mathbf{H}'_i \mathbf{H}'_i{}^T\right)^{-1} \mathbf{H}'_i\right), \quad (5.8)$$

where the operation  $\max(0, \cdot)$  sets all the negative elements of the input matrix to zero. The  $n \times k$  matrix  $\mathbf{H}_1^*$  used in Eq. (5.5) is selected through searching within the

combined set of  $S_1^{(M)} \cup S_0$  based on a predefined score function  $O(\cdot)$  for assessing the cluster quality according to the corresponding encoding coefficient matrices. This is given as

$$\mathbf{H}_1^* = \arg \max_{(\mathbf{W}, \mathbf{H}) \in S_0 \cup S_1^{(M)}} O(\mathbf{H}). \quad (5.9)$$

Letting  $u = \beta \exp^{-\gamma \|\mathbf{A} - \mathbf{H}_i\|_F^2}$ , Eq. (5.6) becomes

$$\mathbf{H}'_i = (1 - u)\mathbf{H}_i + u\mathbf{A}. \quad (5.10)$$

Since  $0 < u \leq 1$ , it is obvious that, when  $\mathbf{H}_i$  and  $\mathbf{A}$  are both non-negative,  $\mathbf{H}'_i$  is non-negative. Also,  $\tilde{\mathbf{W}}_i$  is always nonnegative according to its design. These guarantee that the generated matrix pairs  $(\mathbf{W}'_i, \mathbf{H}'_i)$  are eligible to be used as NMF candidates.

Eq. (5.6) drives those encoding coefficient matrices  $\{\mathbf{H}_i\}_{i=1}^m$  generated by the multiplicative rule to move towards a pre-selected one  $\mathbf{H}_1^*$  that can best serve the clustering task. This design is motivated by a recent evolutionary optimization algorithm inspired by the flashing behaviour of firefly, known as firefly algorithm [112]. The algorithm assumes that attractiveness between fireflies is proportional to their brightness, thus, given any two fireflies, one will move towards the other that glows brighter. However, such attractiveness decreases when the distance between two fireflies increases. Following Eq. (5.6), each candidate in  $S_1^{(M)}$  is viewed as a firefly. The quality of the encoding coefficient matrix for each candidate, evaluated by the score function  $O(\cdot)$ , represents the brightness degree of the firefly. The evolving rule is constructed by letting all the fireflies move towards the brightest one in each iteration. The exponential term  $\beta \exp^{-\gamma \|\mathbf{H}_1^* - \mathbf{H}_i\|_F^2}$  forces the attentiveness towards the brightest firefly to decrease as the relative distance increases. This procedure offers opportunity to evolve higher quality of encoding coefficient matrices to better serve the data clustering task.

Eq. (5.7) computes the basis matrix  $\mathbf{W}'_i$  from the updated encoding coefficient matrix  $\mathbf{H}'_i$ . The design is based on the alternating least squares algorithm for NMF [19, 55, 93], which updates the basis matrix based on the current encoding coefficient matrix through first solving the unconstrained reconstruction error minimization problem of

$$\min_{\mathbf{W}} \left\| \mathbf{X} - \mathbf{W}\mathbf{H}'_i{}^T \right\|_F^2, \quad (5.11)$$

by setting its derivative equal to zero, and then modifying the resulting matrix by converting all its negative elements to zero. This procedure gives the matrix  $\tilde{\mathbf{W}}_i$ . However, the modification step of converting the negative elements to zero potentially raises the risk of obtaining undesired reconstruction error. An alternative setup of  $\mathbf{W}'_i$  is to employ the original one  $\mathbf{W}_i$  as generated by the multiplicative rule, given the fact that the basis matrix does not affect directly the data cluster structure. In Eq. (5.7), we choose between  $\mathbf{W}'_i$  and  $\mathbf{W}_i$  the one possessing the smaller reconstruction error to prevent the proposed evolving procedure from heavy sacrifice of data representation accuracy to compensate the cluster quality.

From the second iteration, the firefly rule starts to create new candidate subset  $S_{t+1}^{(F)}$  by operating on its previously generated subset  $S_t^{(F)}$ . The operation includes two steps, which first modifies  $S_t^{(F)}$  by the multiplicative rule  $\Phi_1$ , and then updates the resulting set based on the firefly updating procedure  $\Phi_2$ . This gives the following new candidate subset for the  $(t + 1)$ th iteration ( $t \geq 0$ ):

$$S_{t+1}^{(F)} = \Phi_2 \left( \Phi_1 \left( S_t^{(F)}, \mathbf{X} \right), \mathbf{H}_t^* \right), \quad (5.12)$$

where

$$\mathbf{H}_t^* = \arg \max_{(\mathbf{W}, \mathbf{H}) \in S_t \cup \Phi_1(S_t, \mathbf{X})} O(\mathbf{H}). \quad (5.13)$$

Here, instead of directly updating  $S_t^{(F)}$  with  $\Phi_2$ , the multiplicative rule is used to smoothen out the given candidates, which may potentially reduce the reconstruction error. To select the best  $\mathbf{H}_t^*$  matrix, a combined set of the previous candidates  $S_t$  and their modified version of  $\Phi_1(S_t, \mathbf{X})$  by the multiplicative rule is used. The used of a mixture of  $\Phi_1$  and  $\Phi_2$  attempts to evolve matrix pairs offering good quality of encoding coefficient matrix while alternatively ensuring the joint quality of the basis and encoding coefficient matrices.

### 5.2.2.3 Survival of the Fittest Rule

This rule ensures the candidates containing the best encoding coefficient matrix are always included in the evolved set. At the first iteration, the candidate subset  $S_1^{(S)}$  is generated by

$$S_1^{(S)} = \Phi_3 \left( S_1^{(M)}, \mathbf{H}_1^* \right), \quad (5.14)$$

After that, it modifies its previously generated subset  $S_t^{(S)}$  by

$$S_{t+1}^{(S)} = \Phi_3 \left( \Phi_1 \left( S_t^{(S)}, \mathbf{X} \right), \mathbf{H}_t^* \right). \quad (5.15)$$

Here, the operation  $S' = \Phi_3(S, \mathbf{A})$  creates  $m + 1$  matrix pairs  $S' = \{(\mathbf{W}'_i, \mathbf{H}'_i)\}_{i=1}^{m+1}$  from the input set  $S = \{(\mathbf{W}_i, \mathbf{H}_i)\}_{i=1}^m$ . It generates the first  $m$  pairs by

$$\mathbf{H}'_i = \mathbf{A}, \quad (5.16)$$

$$\mathbf{W}'_i = \mathbf{W}_i. \quad (5.17)$$

For the last pair, its encoding coefficient matrix is generated by Eq. (5.16), while its basis matrix by

$$\mathbf{W}'_{m+1} = \max \left( 0, \mathbf{X} (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{A} \right). \quad (5.18)$$

This rule combines the best encoding coefficient matrix  $\mathbf{H}_t^*$  selected in each iteration with various basis matrices. Eq. (5.16) can be viewed as a special case of Eq. (5.6) with the fixed parameter  $u = 1$ . This is equivalent to forcing all the weaker fireflies to eliminate themselves but let the brightest one to survive, thus the rule is named as



the survival of the fittest. To combine  $\mathbf{H}_t^*$  with all the possible basis matrices by Eqs. (5.17,5.18), instead of only ones providing smaller reconstruction errors by Eq. (5.7), acts under a similar motivation as that of the genetic operator mutation in evolutionary optimization. It attempts to introduce new candidates by altering the basis matrix of the strongest one to avoid being trapped in a local optimum.

### 5.2.3 Score Function

Since the primary goal of this work is to improve NMF so that it can serve better the data clustering task, it is natural to formulate the score function as a cluster validity measure [25, 62] that assesses the cluster quality. When internal evaluation is used, the assessment is based on the data matrix  $\mathbf{X}$  itself. The cluster structure possessing higher within-cluster similarity and lower between-cluster similarity is of better quality. In this case, the Davies-Bouldin index [21] for example can be directly used as the score function. When external evaluation is used, the assessment compares the clustering results with the ground truth partition of the data, for which the cluster structure that better matches the ground truth partition is of higher quality. Correspondingly, the RAND index [87] for example can be used as the score function, which computes the percentage of the correct data partition offered by the clustering result as compared to the ground truth partition.

The overall data flow of the proposed ENMF is shown in Fig. 5.1. Assuming  $q$  pairs of seed matrices are included in the initial candidate set  $S_0$ , there will be  $q$ ,  $q$  and  $q + 1$  candidates generated respectively in  $S_t^{(M)}$ ,  $S_t^{(F)}$ , and  $S_t^{(S)}$  at each iteration, leading to in total no more than  $3q + 1$  candidates in each iteration as  $S_t^{(M)} \cup S_t^{(F)} \cup S_t^{(S)}$ . It can be seen from Fig. 5.1 that the operation  $\Phi_1$  based on the multiplicative rule is used to bridge between iterations aiming at generating candidates that are able to converge to an NMF solution driven by the reconstruction error minimization. Additionally, the operations of  $\Phi_2$  and  $\Phi_3$  based on the firefly and survival of the fittest rules aim at the local improvement of the candidates to produce better quality of data clusters within each iteration. Specifically,  $\Phi_2$  moves the the encoding coefficient matrices generated by the multiplicative rule towards a best one selected based on a pre-defined score function, while  $\Phi_3$  ensures the best selected encoding coefficient matrix is included in the updated candidate set. The score function assesses the cluster quality so that the output of NMF is able to serve better the data clustering task.

## 5.3 Experimental results and analysis

Experimental evaluation is conducted with nine benchmark classification datasets from UCI machine learning repository, including balance scale, breast tissue, breast cancer Wisconsin diagnostic (WDBC or cancer), breast cancer Wisconsin original (BCWO

Table 5.1: Details of the datasets used.

Datasets	Instances ( $n$ )	Features( $d$ )	Classes ( $k$ )
Balance	625	4	3
Breast	106	9	6
WDBC	569	30	2
BCWO	683	9	2
Dermatology	358	34	6
Glass	214	9	6
Haberman	306	3	2
Iris	150	4	3
Thyroid	215	5	3

Table 5.2: Performance comparison for different datasets in RAND, which is reported in percentage (%). The best performance is highlighted in bold, and the second best is underlined.

Method	Balance	Breast	WDBC	BCWO	Dermatology	Glass	Haberman	Iris	Thyroid
K-means	59.7	71.9	75.2	92.4	70.5	69.0	50.5	83.3	74.1
FCM	58.6	66.3	75.2	91.6	70.8	71.8	50.7	88.3	72.6
NMF-CI1	63.8	72.0	75.2	92.4	71.1	70.4	50.4	88.2	78.1
NMF-CI2	65.9	68.7	75.2	91.6	70.8	71.8	50.7	88.3	73.6
NMF-CI3	65.6	69.6	72.7	63.9	80.7	<u>72.5</u>	50.9	<u>93.5</u>	76.4
NMF-RI	65.7	64.5	71.1	68.3	85.6	69.5	50.4	81.5	76.2
NMF-RAI	64.8	57.6	69.2	68.0	86.1	69.4	51.9	81.1	74.6
NMF-MIX	65.9	71.4	75.2	92.4	85.9	<u>72.5</u>	51.8	<u>93.5</u>	78.2
ENMF-CI1	66.0	72.0	75.2	92.4	73.3	70.7	50.4	88.2	78.1
ENMF-CI2	67.3	68.7	75.2	91.6	70.8	71.8	50.7	88.3	73.6
ENMF-CI3	<b>81.2</b>	67.9	<u>76.7</u>	<b>93.5</b>	82.0	72.3	50.5	<u>93.5</u>	<b>83.5</b>
ENMF-RI	73.9	<b>75.2</b>	74.3	71.1	85.5	69.7	<u>54.7</u>	86.2	79.5
ENMF-RAI	73.2	<u>73.9</u>	69.3	70.1	<b>86.9</b>	69.3	54.0	85.6	77.5
ENMF-MIX	<u>76.0</u>	72.9	<b>77.1</b>	<b>93.5</b>	<u>86.2</u>	<b>72.7</b>	<b>55.0</b>	<b>95.6</b>	<b>83.5</b>

or cancer-int), dermatology, glass identification, Haberman’s survival, iris and thyroid disease. The characteristics of all the datasets are summarised in Table 5.1, where the first word or the abbreviation of the data name are used to refer each dataset. The clustering performance is assessed through external evaluation, comparing the computed cluster partition with the ground truth class partition based on the RAND index. For data preprocessing, a scalar  $\|\min x_{ij}\|$  is added to the input data matrix  $\mathbf{X}$  when it contains negative elements.

### 5.3.1 Experimental Setup

In order to thoroughly and fairly compare our proposed NMF updating strategy with the state-of-the-art one, meanwhile to investigate effects of different NMF initialization approaches, the following experiments are conducted.

To compare the proposed ENMF with the classical multiplicative update [24], the

Table 5.3: Performance improvement of ENFM over NMF under different initialization approaches.

	Balance	Breast	WDBC	BCWO	Dermatology	Glass	Haberman	Iris	Thyroid	t-test
CI1	+2.2%	0%	0%	0%	+2.2%	+0.3%	0%	0%	0%	0.0898
CI2	+1.4%	0%	0%	0%	0%	0%	0%	0%	0%	0.0275
CI3	+15.6%	-1.7%	+4.0%	+29.6%	+1.3%	-0.2%	-0.4%	0%	+7.1%	1.0336
RI	+8.2%	+10.7%	+3.2%	+2.8%	-0.1%	+0.2%	+4.3%	+4.7%	+3.3%	0.8901
RAI	+8.4%	+16.3%	+0.1%	+2.1%	+0.8%	-0.1%	+2.1%	+4.5%	+2.9%	0.8534
MIX	+10.1%	+1.5%	+1.9%	+1.1%	+0.3%	+0.2%	+3.2%	+2.1%	+5.3%	0.4732

results are examined under each of the five initialization approaches including RI, RAI and three types of CI based on the membership matrix of k-means (CI1), the membership matrix of FCM (CI2) and the member degree matrix of FCM (CI3). Every initialization approach is run five times, generating five pairs of encoding coefficient and basis matrices. For ENMF, these five pairs are used as the initial candidates to start the algorithm. By splitting the dataset into two separate sets for training and test purposes, the RAND index computed with the training set is used as the the score function to drive the evolving of ENMF, while the RAND index computed with the test set is used to report the performance. For standard NMF based on multiplicative update, the same five matrix pairs are used to initialize the updating procedure, leading to five solutions of NMF. The one possessing the highest RAND index computed with the training data is selected to report the RAND performance with the test data. In order to compare the general performance, five runs of k-fold ( $k = 4$ ) cross validation (CV) has be performed, which repeats the aforementioned procedure  $4 \times 5 = 20$  times based on different random training-test partitions of the data. In the end, an average RAND index is computed over the 20 sets of test data and is reported as the final performance. In the end, the results are also examined using a mixture of all the five initialization approaches (MIX). Five pairs of initial coefficient and basis matrices are generated by running each of the five initialization approaches once. Then, the same procedure as described above is used to evaluate the ENMF and NMF. The number of basis vectors  $k$  is set as the cluster number for both NMF and ENMF. The ENMF parameters are set as  $\gamma = \frac{1}{\max_i \|\mathbf{A} - \mathbf{H}_i\|_F^2}$  and  $\beta = 1$ . The iteration numbers for NMF and ENMF updates are both fixed as 500 in all experiments.

### 5.3.2 Results and Analysis

Table 5.2 compares the NMF with multiplicative update and the ENMF with the proposed evolutionary update, under different initialisation setups of CI1, CI2, CI3, RI, RAI and MIX as explained in previous section. It can be seen that the best performance for each dataset is always obtained by ENMF. For more clear identification of performance improvement of ENMF over NMF, we summarize in Table 5.3 the

performance difference of ENMF–NMF under different initialization setups. It can be seen that, in Table 5.3, out of the 60 entries, there are 5 negative ones, 16 zeros and 39 positive ones. For 65% of the cases, ENMF performs better than NMF by between 0.2% and 29.6%, for 27% of the cases, the two methods achieve the same performance, while for 8% of the cases, ENMF performs worse than NMF by between 1.7% and 0.1%. For some cases, ENMF significantly improves NMF by over 8%. We also calculated the t-test values to illustrate the significant difference between NMF and ENMF with CI1-3, RI, RAI and MIX. It shows that ENMF with CI3, RI, RAI and MIX have the significant difference according to the large t-test. Among them, ENMF with MIX has the positive performance improvement for all the datasets. Therefore, we can conclude that ENMF with MIX significantly outperforms the NMF with MIX for these datasets.

As shown in Table 5.2, performance of ENMF and NMF initialized with one single approach is often affected by the used approach. There is no superiority of one initialization approach over the other given different datasets. This leaves extra effort to the user in terms of choosing an appropriate initialization setup. A mixed initialization that automatically takes advantage of different approaches is used for ENMF, which leads to better performance for most datasets. This shows the advantages of hybridization. When ENMF employs similar seed matrices, e.g., those obtained by CI1 and CI2, they do not motivate the evolving procedure to generate better candidates than NMF, exemplified by those zero improvements in Table 5.3. Differently, random initialization, such as RAI and RI, offers solutions of varying quality. They help to avoid the local optimum by preventing the generation of candidates that are too similar to each other, however, may lead to unsatisfactory convergence without sufficient number of iterations due to its weakness at seed quality control. By initialize the ENMF with a mixture of different types of seed matrices, their quality and diversity are balanced, thus are able to evolve solutions possessing more significant performance improvement, as shown in the last row of Table 5.3.

In Figure 5.2–5.10, we compare convergence of ENMF based on the mixed initialization using RI, RAI, CI1, CI2 and CI3 and the NMF multiplicative update initialized by different approaches of RI, RAI, and the best one from CI1 to CI3 (referred as CI in the figure), for different datasets. The RAND performance is computed with the test data. It can be seen that ENMF always provides the best clustering performance, meanwhile it offers significantly faster convergence rate than NMF with RI and RAI for many datasets. It is observed that, for NMF initialized with the output of a clustering algorithm, the quality of the clusters indicated by the encoding coefficient matrix  $\mathbf{H}$  never improves over the update for almost all the datasets apart from balance and dermatology. The combination of NMF and clustering based initialization is only worthy when the later NMF update is able to improve the cluster quality. Otherwise, one can just directly apply the clustering algorithm on its own to save the extra computational

cost consumed in the NMF update. Differently, ENMF initialized by a mixture of different approaches significantly improves the clustering performance over iterations and also converges fast for almost all the datasets.

Given the input data, ENMF evolves the factorization matrices that best serve the clustering task, by considering both the reconstruction error and cluster quality within its update. Starting from  $q$  pairs of seed matrices, no more than  $3q + 1$  candidates are generated by ENMF in each iteration. Correspondingly, NMF with multiplicative update that selects the best solution given  $q$  different initialization matrices requires the generation of  $q$  pairs of  $\mathbf{W}$  and  $\mathbf{H}$  matrices in each iteration. Among the ENMF candidates, the  $q$  candidates generated by the multiplicative rule consumes exactly the same cost as that by NMF. The additional cost of ENMF is consumed by the those candidates generated by the firefly and survival of the fittest rules.

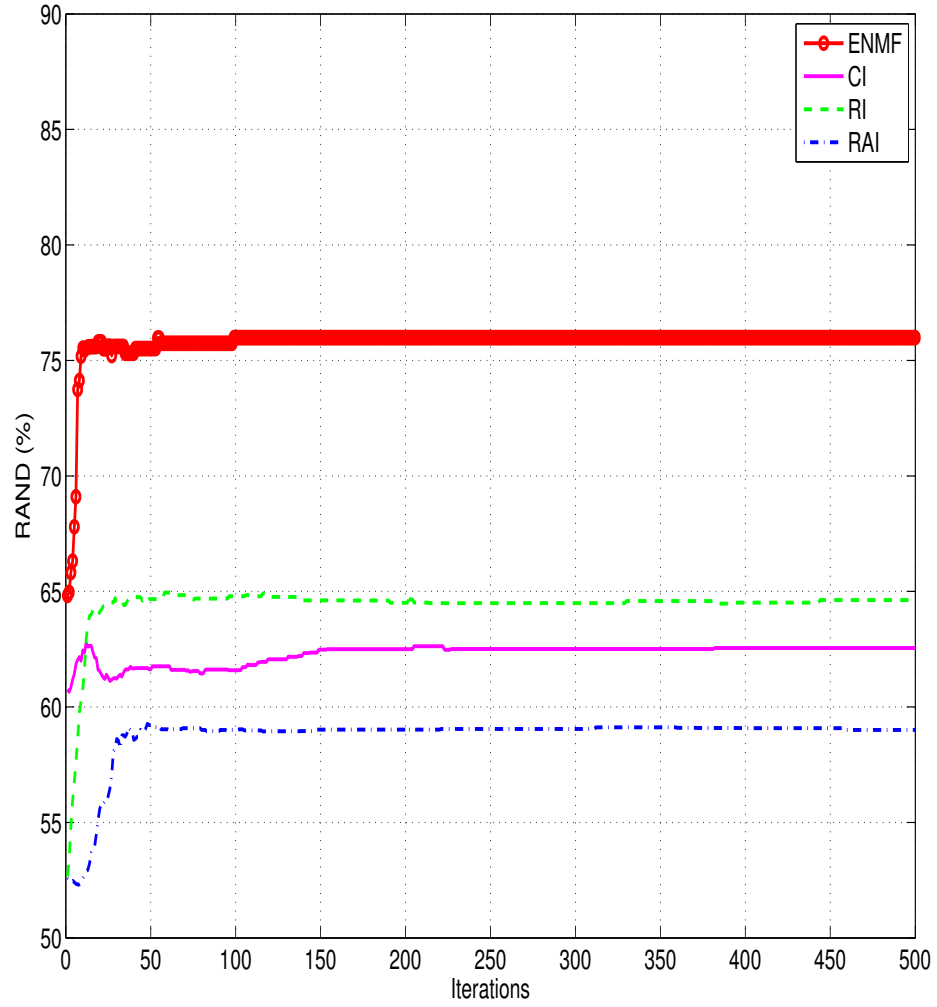


Figure 5.2: The accuracies of the corresponding testing dataset for balance (ENMF-MIX)

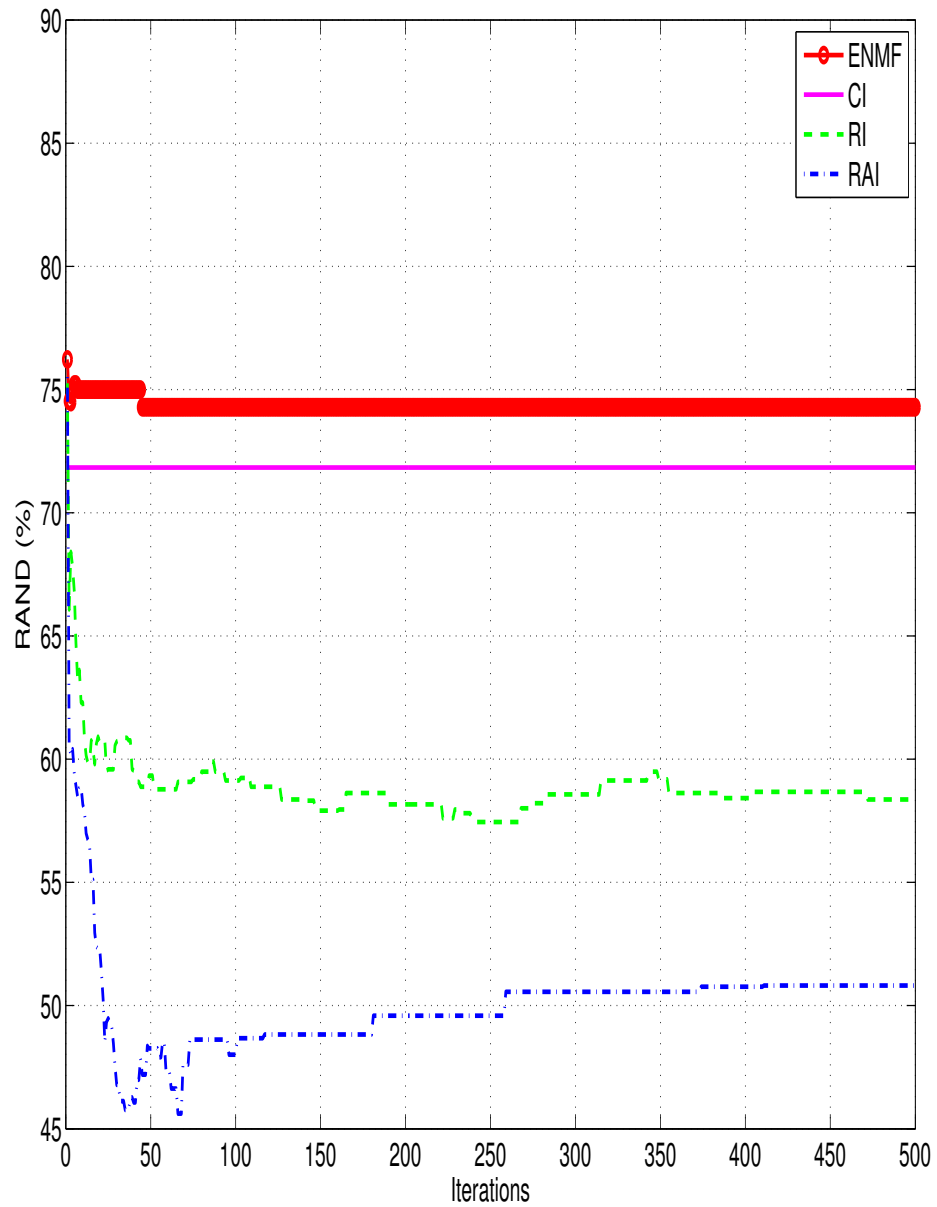


Figure 5.3: The accuracies of the corresponding testing dataset for breast tissue (ENMF-MIX)

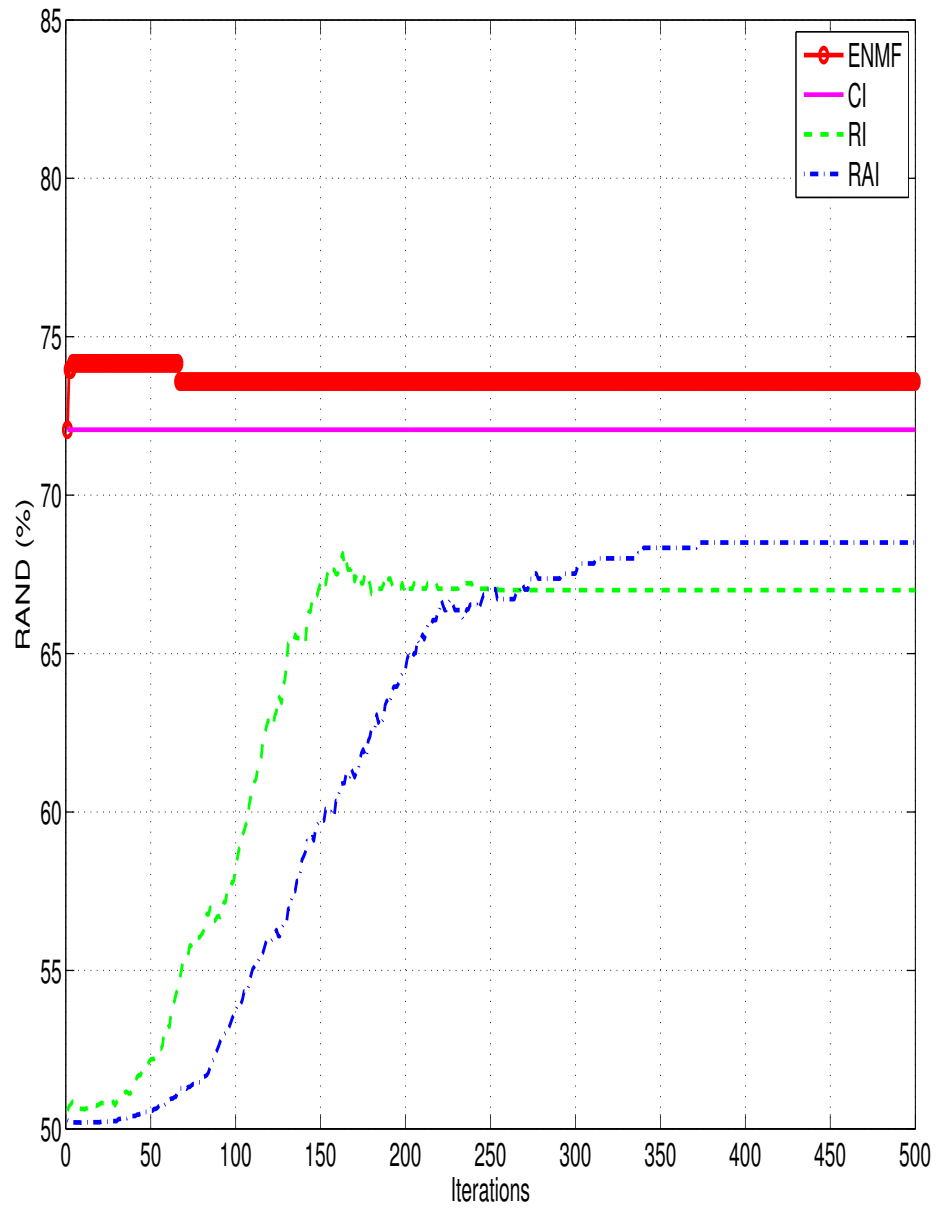


Figure 5.4: The accuracies of the corresponding testing dataset for cancer (ENMF-MIX)



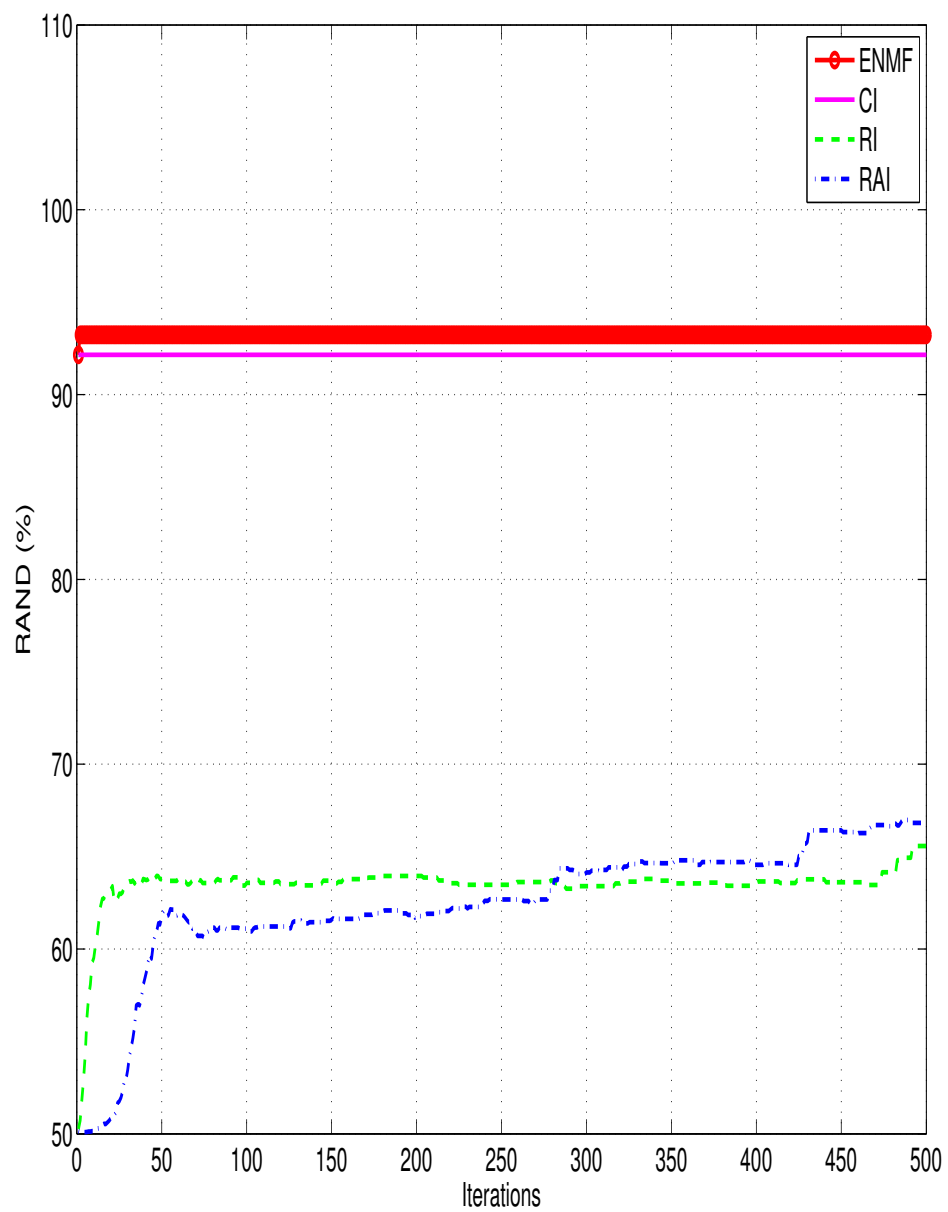


Figure 5.5: The accuracies of the corresponding testing dataset for cancerint (ENMF-MIX)

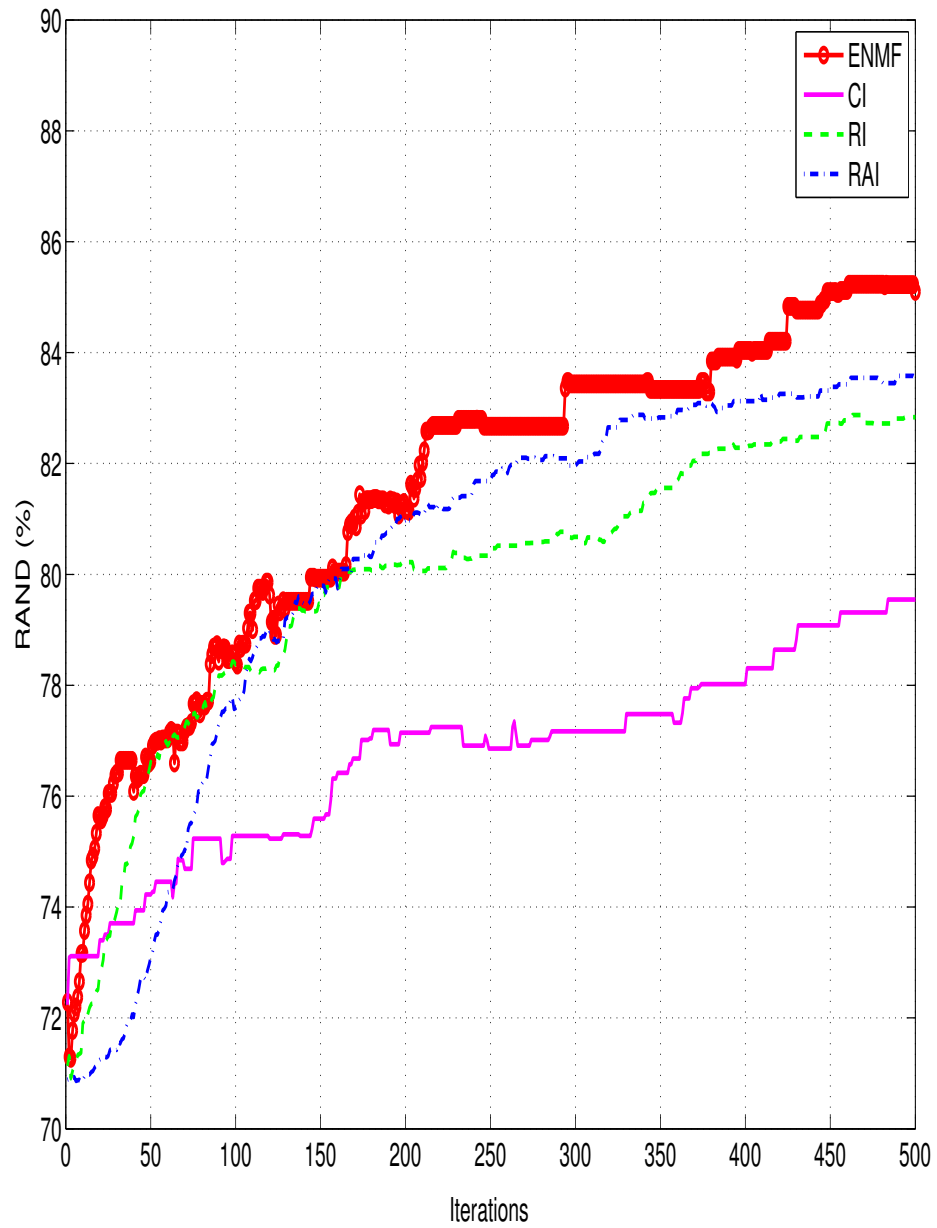


Figure 5.6: The accuracies of the corresponding testing dataset for dermatology (ENMF-MIX)

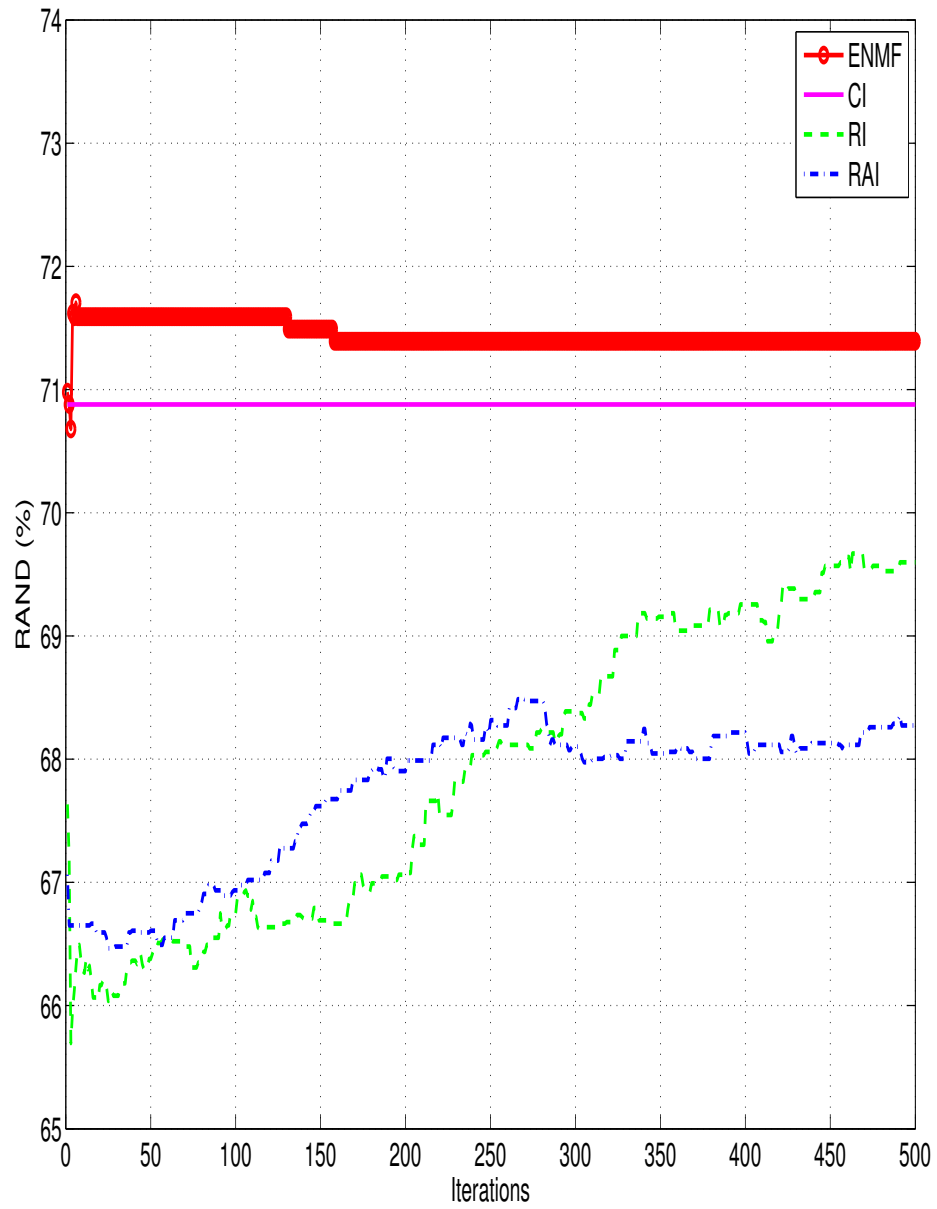


Figure 5.7: The accuracies of the corresponding testing dataset for glass (ENMF-MIX)

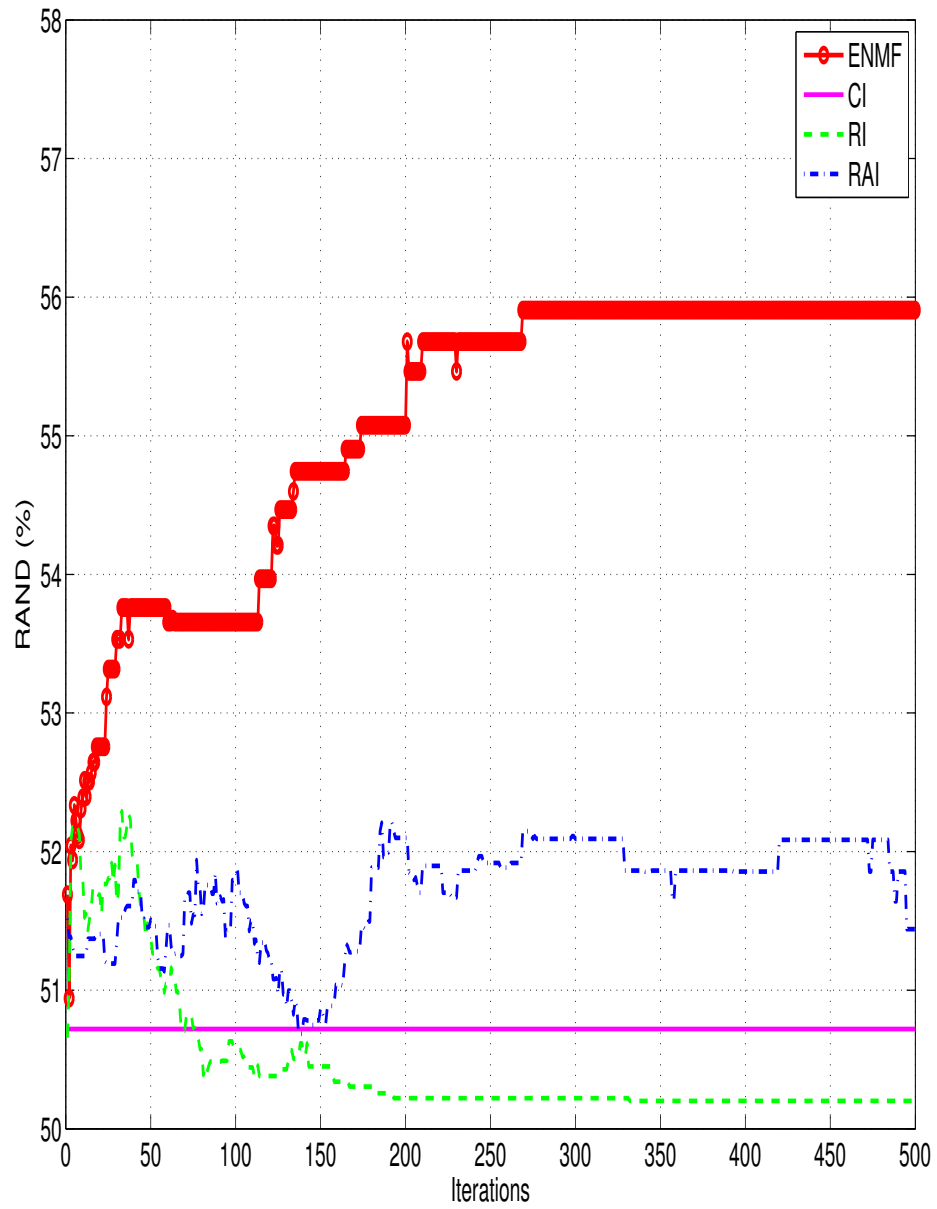


Figure 5.8: The accuracies of the corresponding testing dataset for haberman (ENMF-MIX)

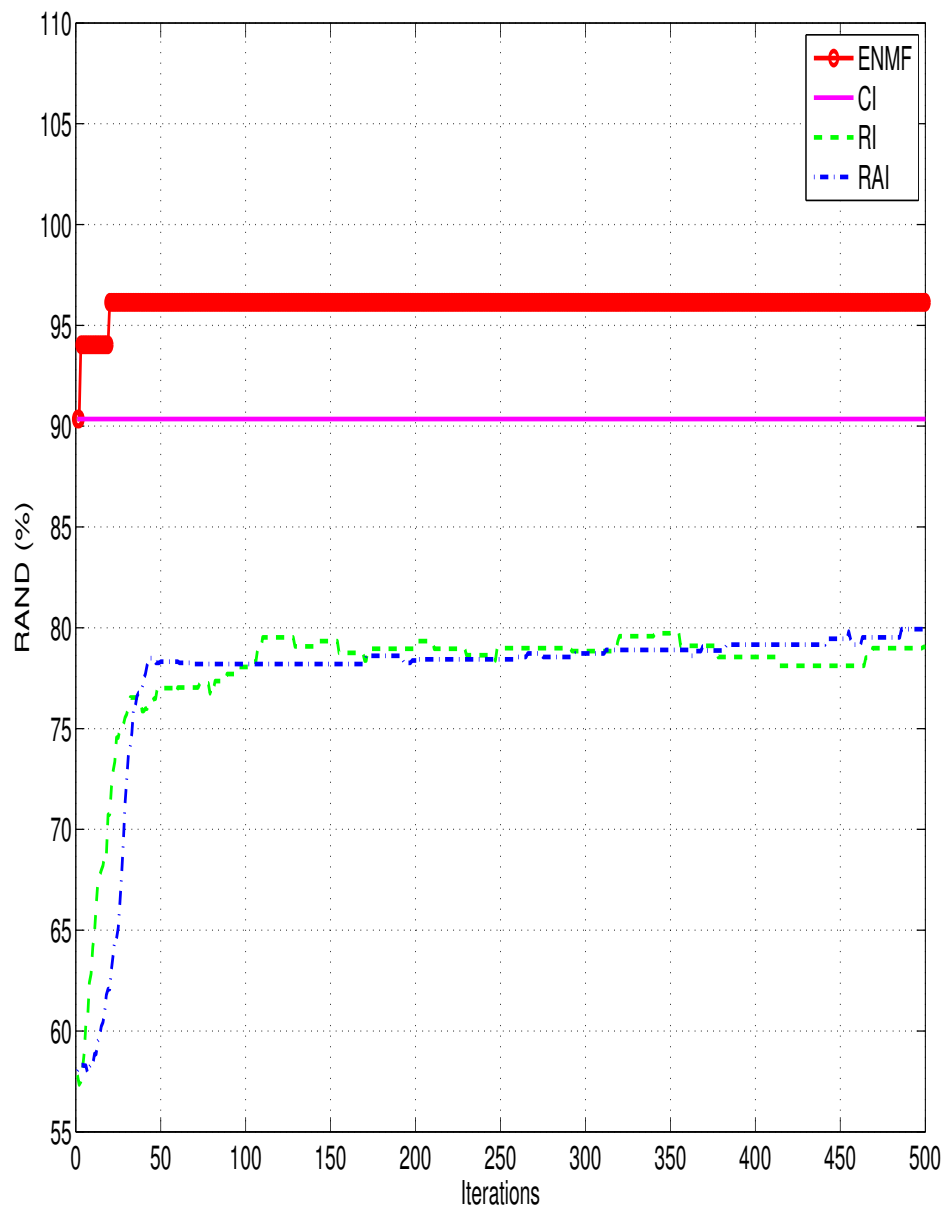


Figure 5.9: The accuracies of the corresponding testing dataset for iris (ENMF-MIX)

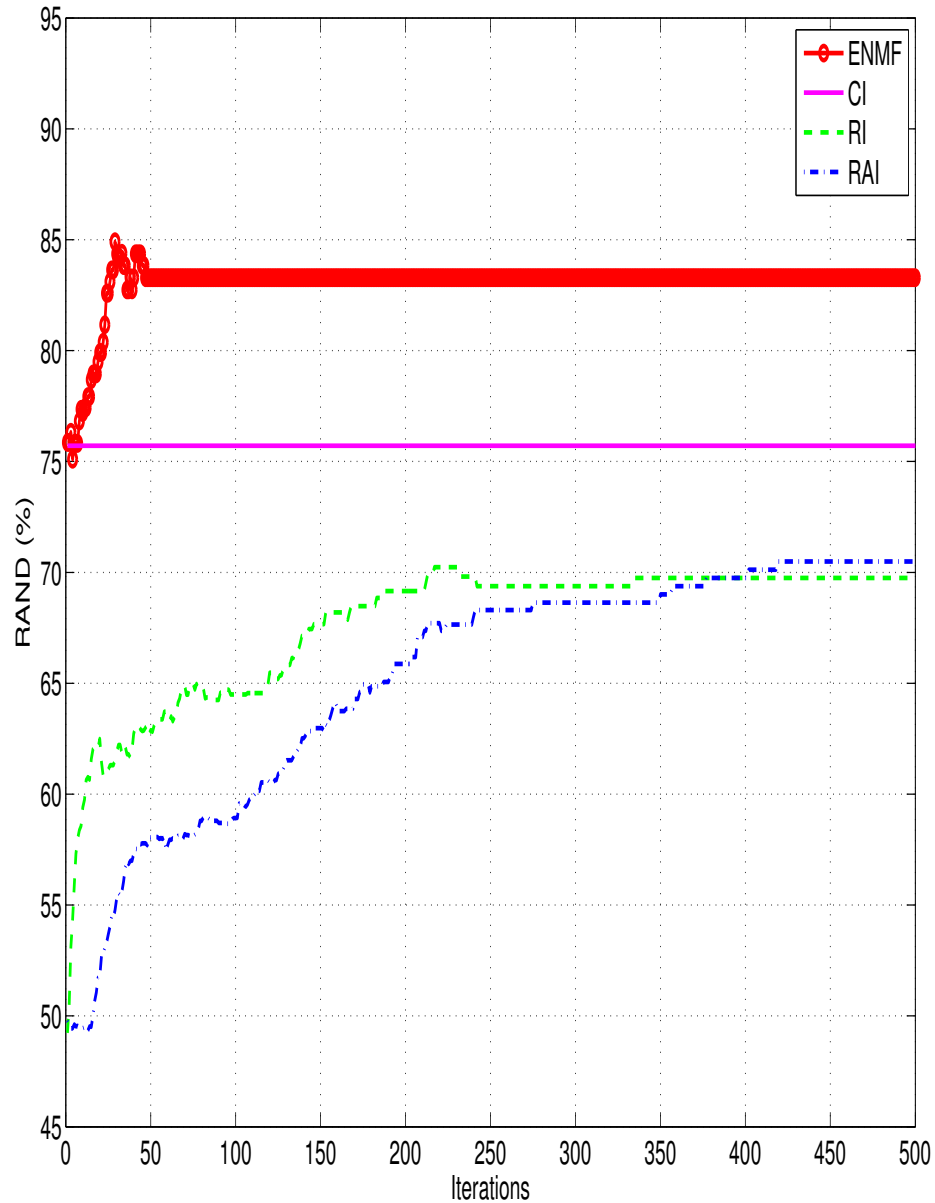


Figure 5.10: The accuracies of the corresponding testing dataset for thyroid (ENMF-MIX)

## 5.4 Conclusion

We have proposed a new strategy for conducting NMF so that the resulting encoding coefficient matrix  $\mathbf{H}$  is capable of representing better quality of data cluster structure. Three rules have been designed, of which the first rule inherits the classical multiplicative update, while the other two rules are driven by preserving stronger candidates

offering higher quality of clusters and meanwhile incorporating their altered versions to avoid local optimum, inspired by the evolutionary optimization algorithm of firefly. Any measure for assessing clustering performance can be used as the score function to drive the evolving procedure. The proposed framework is a very general one, which can also be applied to improve NMF applications to other data analysis task by setting appropriate score function. For example, measures of compression rate, data sparsity and reconstruction error can be used for data compression task. Experimental results have demonstrated the superior performance of the proposed method over the existing ones for data clustering evaluated with nine benchmark datasets.

## Chapter 6

# Proposed NMF Updating Strategy for Data Compression

This chapter propose an evolutionary optimization strategy for NMF to the image datasets driven by three proposed movement schemes in the solution space, saying original movement, beta movement and best movement. Since we aim at producing NMF output that facilitates data compression, these movements are guided by the objective functions that makes use of the compression quality measurements saying sparsity, orthogonality and error. To increase the rate of convergence, a hybrid initialisation approach is used by including the state-of-the-art NMF initialization methods as seed knowledge, such as k-means based initialization and fuzzy c-means based initialization. These are to be combined with multiple types of random initialization. There is no limitation for the number and the type of the initialization methods used for the proposed optimisation approach. Experiments were carried out using Yale and ORL image datasets and the results have shown that the proposed NMF evolutionary optimization strategy significantly improves the data compression performance as compared to the other existing initialized NMF methods. Work of this chapter is prepared as the following paper.

Liyun Gong, T. Mu and Al-Nuaimy Waleed, Evolutionary nonnegative matrix factorization for data compression, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2015. (to be submitted)

### 6.1 Introduction

In this chapter, an evolutionary optimization strategy based on NMF is proposed as the main core to solve the data compression problem. Based on the traditional NMF, the initial values of the parameter of NMF are set as random values. However, this is not the most effective setup. Many algorithms have been proposed to obtain the initial values of  $W$  and  $H$  in a more sophisticated way to improve the rate of convergence. With these initialization methods, enhanced convergence rates as well as better accu-



racy can be achieved.

Due to the sensitivity to the initial values of parameters of NMF from the different initialization methods, different compression results may be obtained for a given dataset. It is challenging to choose an appropriate NMF initialization scheme for a given data compression task. Instead of choosing one particular initialization scheme, we propose a different NMF algorithm that allows an evolutionary optimisation procedure, starting from different initialization candidates in the solution space and effectively converging along multiple directions, in order to obtain better quality of data compression. The proposed evolutionary optimisation strategy is general, and there is no limitation for the number and the type of the initialization methods used in the optimization. About evolutionary optimisation, there has been a rich literature over the past decades. One classical method is genetic algorithm (GA) [96, 108] that solves search and optimization problem by the principles of natural evolution. It starts with a random population of solutions. The solution evolution is then processed through successive iterations by the fitness evaluation such as crossover and mutation operation. At the end of the evolution, the algorithms converge to the best solution, which may represent the optimal solution to the given problem. Firefly algorithm (FA) is another method which has been proposed by Yang et al ([112]). recently. Different with GA, this algorithm is to find the optimal solution of the problem based on the behaviour of fireflies. Each firefly represents a solution to a specific optimization problem, and it is attracted by the other brighter neighbouring fireflies at each iteration step. The brightness is defined by the objective function and the attractiveness decreases as their distance increases.

The proposed evolutionary optimization strategy takes advantages of properties of both GA and FA. It selects the superior and eliminates the inferior as that of GA, while allows each candidate solution to move along a specific direction by following the FA rule. In each iteration, the candidate updating rules are mainly driven by the classical NMF updates with the consideration of data compression quality measure. Such a design leads to not only good convergence of NMF but also good quality of data compression. The experiments were carried out on Yale and ORL image datasets and we compare our proposed optimization strategy with some standard initialized NMF methods by sparsity, orthogonality, error and RAND index.

The rest of the chapter is organized as follows. In Section 6.2, the proposed optimization strategy in the field of data compression is described. Experimental results based on these methods are evaluated and analyzed in Section 6.3. Finally, Conclusion is drawn at the end of this chapter.

## 6.2 Proposed evolutionary optimization strategy on data compression

Here we proposed an evolutionary optimization strategy to improve the data compression performance of the initialized NMF methods. Each pair of initial factors ( $\mathbf{W}$ ,  $\mathbf{H}$ ) obtained from each initialization method is updated through successive iterations (generations) under the support of NMF method and three proposed movement methods, saying original movement, beta movement and best movement. After several generations of evolution, the algorithm converges to the best solution which has the best data compression performance. The general structure of the proposed methods is shown in figure 6.1.

Assume that  $q$  initialization methods as the seeds are applied to our proposed method evolutionary optimization strategy. "x" here represents one pair of the initial factors ( $\mathbf{W}$ ,  $\mathbf{H}$ ) obtained from one of  $q$  initialization methods. "o", " $\Delta$ " and " $\square$ " represents the pairs of the factors ( $\mathbf{W}$ ,  $\mathbf{H}$ ) moved under original movement, best movement and beta movement process separately. We first obtain  $q$  pairs of the initial factors ( $\mathbf{W}$ ,  $\mathbf{H}$ ) from  $q$  different initialization methods. Then there are two main processes at each iteration for the proposed evolutionary optimization strategy. One process is to apply multiplication update rule once to all the factors by equation 3.3. That is, to update each factor once by NMF method and it is followed by applying the three proposed movement methods to the factors to update the current  $\mathbf{W}$  solutions. Our proposed method is different from GA with the fitness evolution which is designed to be more suitable for NMF update in data compression area compared with crossover and mutation in GA. Also the property of the movement method in FA is inducted and re-designed to be available for the beta movement method. The flow chart of our proposed evolutionary optimization method is shown in figure 6.2. In this chapter, we explore the data compression performance of the initialized NMF which is associated to the factor  $\mathbf{W}$ . So the movement methods are applied to  $\mathbf{W}$  value while keeping the same value for  $\mathbf{H}$  during the process. We describe each proposed movement method for  $\mathbf{W}$  in details in the following sections.

<b>0<sup>th</sup> Iteration</b>	<b>1<sup>st</sup> Iteration</b>		<b>2<sup>nd</sup> Iteration</b>		<b>t<sup>th</sup> Iteration</b>	
$(q)$	NMF $(q)$	Move $(3q)$	NMF $(3q)$	Move $(3q)$	NMF $(3q)$	Move $(3q)$
X	0	0	0	0	0	0
X	0	0	0	0	0	0
X	0	0	0	0	0	0
...	...	...	...	...	...	...
X	0	0	0	0	0	0
		Δ	Δ	Δ	Δ	Δ
		Δ	Δ	Δ	Δ	Δ
		Δ	Δ	Δ	Δ	Δ
		...	...	...	...	...
		Δ	Δ	Δ	Δ	Δ
		□	□	□	□	□
		□	□	□	□	□
		□	□	□	□	□
		...	...	...	...	...
		□	□	□	□	□

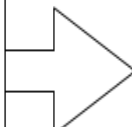


Figure 6.1: The general process of the proposed evolutionary optimization strategy

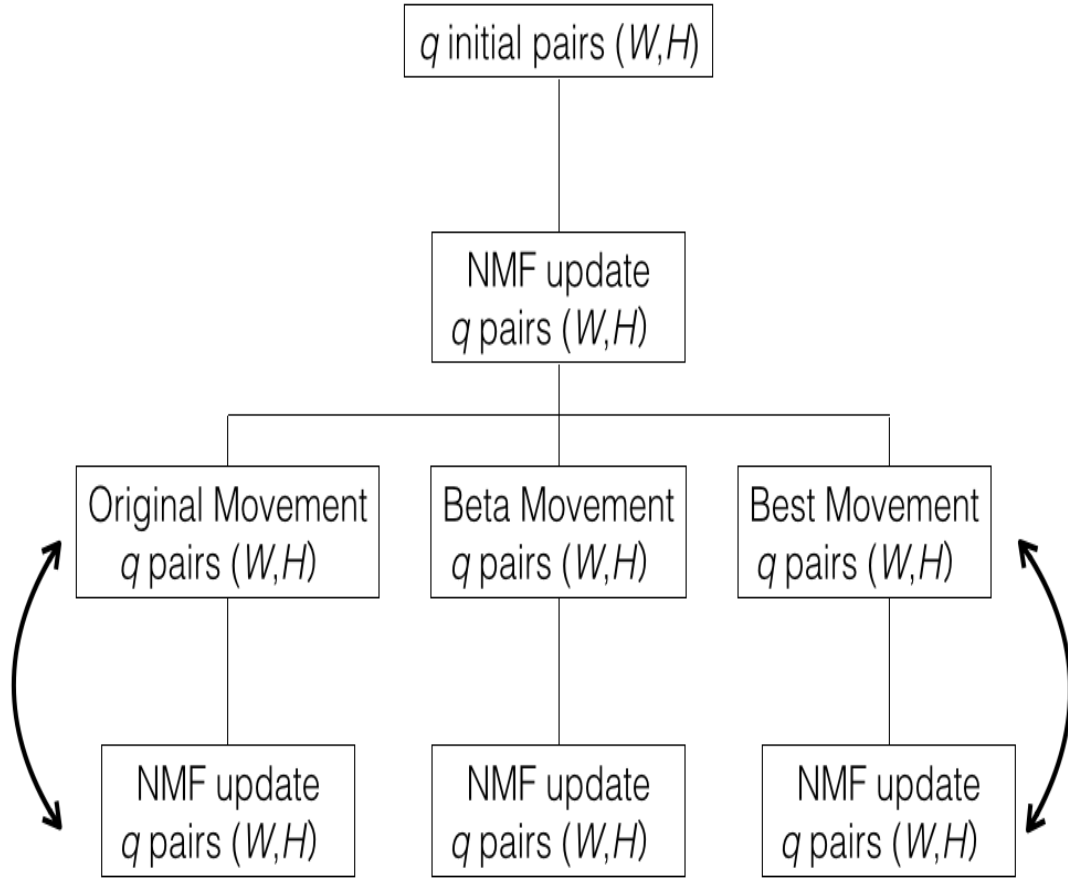


Figure 6.2: The flow chart of the proposed evolutionary optimization strategy

### 6.2.1 Original movement for $\mathbf{W}$

The original movement is the basic movement which just keeps the original factors from the previous step. That is, there is mathematically no movement for the factors during this movement. The process is supervised by "o" in figure 6.1. It starts with  $q$  pairs of the initialized factors  $(\mathbf{W}, \mathbf{H})$  from  $q$  initialization methods. At each iteration, NMF update rule is applied once to all these pairs according to the equation 3.3 and then no movement after that. This process is the same as the NMF method which starts by the same pairs of initializing factors  $(\mathbf{W}, \mathbf{H})$  and these pairs are iteratively updated to minimize the equation 3.2. For each step, we find one best  $\mathbf{W}$  solution which has the highest compression performance according to the objective function from three proposed movement methods. The original movement is added in the process so that the basic initialized NMF is always taken into the consideration. That is, the compression performance of the initialized NMF is kept by the original movement while the other two movements are applied to re-arrange the factors. The best  $\mathbf{W}$  solution will be

found at each iteration which is always equal or even better than the solutions from the basic initialized NMF. As many initialization methods for NMF has been proposed in literature, choosing the best one for a given dataset would be the challenging task. Our proposed original movement can solve the problem by automatically selecting the best solution among several initialization methods.

### 6.2.2 Beta movement for $\mathbf{W}$

During each step of FA, for any two solutions, one solution will move towards to the other solution that has the better objective value. Here we select this property and make modification on it to match our evolutionary optimization strategy method. The proposed beta movement is shown as follows.

$$\mathbf{W}_s^{t\beta} = \mathbf{W}_s^t + \beta(\mathbf{W}^{t*} - \mathbf{W}_s^t) \quad (6.1)$$

where  $\beta \in [0, 1]$  and  $s \in [0, q]$ . Here  $\mathbf{W}_s^t$  is the  $\mathbf{W}$  factor from  $s^{th}$  initialization method after  $t^{th}$  NMF update during beta movement process.  $\mathbf{W}^{t*}$  is the best  $\mathbf{W}$  solution found at  $t^{th}$  iteration which is described in section 6.2.3. In order to reduce the computation complexity, the previous  $\mathbf{W}$  factors  $\mathbf{W}_s^t$  are collected only from the previous beta movement process. That is, the symbol "□" after  $t^{th}$  NMF update and just before the movement process. According to this formula, the new factor  $\mathbf{W}_s^{t\beta}$  is computed by moving the previous value  $\mathbf{W}_s^t$  towards the current best solution  $\mathbf{W}^{t*}$  with the certain distance determined by the coefficient  $\beta$ . The selection of the value  $\beta$  here depends on the experience. As shown in equation 6.2, the  $\mathbf{W}$  value always keep the same value as the previous one when  $\beta$  is set to be zero which is equivalent to the original movement. However, when  $\beta$  is one, the  $\mathbf{W}$  value changes directly to the current best  $\mathbf{W}$  which is the same as the best movement described later. The value  $\beta$  that is too high may lead to premature convergence of the algorithm and loss of good solutions. Here we test the datasets at  $\beta = 0.1$  to observe the performance.

$$\mathbf{W}_s^{t\beta} = \begin{cases} \mathbf{W}_s^t & \text{if } \beta = 0 \\ \mathbf{W}^{t*} & \text{if } \beta = 1 \end{cases} \quad (6.2)$$

With original and best movement included, we can only choose the best  $\mathbf{W}$  solution from a number of the initialized NMF. It makes no difference to apply initialization methods to NMF separately and select the best solution among them for a given dataset. However, with the addition of the beta movement, we move the factor  $\mathbf{W}$  by 10 percentage ( $\beta = 0.1$ ) of the distance between the  $\mathbf{W}$  value and the current best  $\mathbf{W}$  position to search more significant position which may has the good or even better compression performance.

### 6.2.3 Best movement for $\mathbf{W}$

Best movement is to move the  $\mathbf{W}$  values directly to the current best  $\mathbf{W}^*$  while keeping the same values for  $\mathbf{H}$ . The best  $\mathbf{W}^*$  in this chapter is determined by the dif-

Table 6.1: The procedure for finding the best solution  $\mathbf{W}^{t*}$

<p><b>STEP1:</b> We have <math>(\mathbf{W}^{rank1}, \mathbf{W}^{rank2}, \dots, \mathbf{W}^{rankh}, \dots, \mathbf{W}^{rank(2qor6q)})</math></p> <p><b>STEP2:</b> Apply non-negative least square (NNLS) to <math>\mathbf{W}^{rank1}</math> and calculate the <math>\mathbf{H}^{rank1}</math> value according to the values of <math>\mathbf{W}^{rank1}</math> and the image dataset.</p> <p><b>STEP3:</b> Calculate the error for the pair <math>(\mathbf{W}^{rank1}, \mathbf{H}^{rank1})</math>.</p> $error = \frac{\ A - \mathbf{W}^{rank1}\mathbf{H}^{rank1}\ }{\ A\ } \times 100\% \quad (6.3)$ <p><b>STEP4:</b> Compare the error value obtained in equation 6.3 with <math>error</math> which is predefined by user. If <math>error \leq error</math>, the best solution <math>\mathbf{W}^{t*}</math> is equal to <math>\mathbf{W}^{rank1}</math>. However, if <math>error &gt; error</math>, repeat STEP2 to STEP4 for <math>\mathbf{W}^{rankh}</math>, <math>h = 1, 2, \dots, 2qor6q</math></p> <p><b>STEP5:</b> If the error of the last <math>\mathbf{W}</math> value <math>\mathbf{W}^{rank(2q/6q)}</math> is still bigger than <math>error</math>, then <math>\mathbf{W}^{t*} = \mathbf{W}^{rank1}</math></p>
---

ferent objective functions which are the set of the data compression quality measurements and described in section 6.2.4. We first collect all the  $\mathbf{W}$  values not only after but also before each NMF update process. It can be seen in figure 6.1 that  $2q$   $\mathbf{W}$  values are collected at the first iteration while  $6q$   $\mathbf{W}$  values are taken into account starting from the second iteration. Then the objective values of these  $\mathbf{W}$  factors are calculated so that  $\mathbf{W}$  factors are recorded in the increasing order shown as  $(\mathbf{W}^{rank1}, \mathbf{W}^{rank2}, \dots, \mathbf{W}^{rankh}, \dots, \mathbf{W}^{rank(2q/6q)})$ . Here  $\mathbf{W}^{rankh}$ ,  $h = 1, 2, \dots, 2q/6q$  is the factor  $\mathbf{W}$  which has the top  $h^{th}$  objective value. In order to obtain the best solution  $\mathbf{W}^{t*}$  which has the best data compression performance and also whose error is within the range predefined by users, we create the process as shown in table 6.1. Nonnegative least square (NNLS) in this table has been proposed by Lawson and Hanson ([?]). It gives the least squares solution to  $\|\mathbf{WH} - \mathbf{X}\|_F^2$  subject to the additional constraint  $\mathbf{H}_{i,j} \geq 0$ . Here we only concerned the least squares solution  $\mathbf{H}$  which also satisfies the NMF nonnegative constraints.

In order to match the fast convergence and the best data compression performance, the changing of  $\mathbf{W}$  is made during each movement process which may lead to the unexpected error. Here the parameter  $error$  is added in the best movement process to satisfy the user's predefined error range. We also have considered the situation that the error of all the factor  $\mathbf{W}$  may be still bigger than  $error$  at some iterations. For this situation, we choose the best  $\mathbf{W}$  value which has the best objective value without considering the error value (shown in STEP5). Here we introduce a parameter called  $minerror$ . It is possible that the error can exceed the the predefined error range because of STEP5 in table 6.1. So the  $minerror$  is defined as the minimum  $error$  value that makes the error within the predefined error range. The main goal of the NMF update rule is to minimize the difference (equation 3.2) and the compression performance may reduced after each NMF update process. So we also consider the  $\mathbf{W}$  values before each

NMF update to make sure that the compression performance of  $\mathbf{W}$  always keeping the same or increasing during each iteration of the proposed evolutionary optimisation strategy. There is one best solution  $\mathbf{W}^*$  and  $q$  different values of  $\mathbf{H}$  factors at each step. So we obtain  $q$  different pairs of "□" for each step shown in figure 6.1. Using the best movement, not only one new root is added to search the best solution, but also the current best solution  $\mathbf{W}^*$  is found and recorded during the iterations.

#### 6.2.4 Objective function

Objective function is an important core in our proposed method which can decide what kind of the performance can be improved. Here we analyze our method with emphasis on the data compression performance including sparsity and orthogonality. In order to improve their performance, there are three types of the objective functions including sparsity, orthogonality and their combination. Here each basis image in  $\mathbf{W}$  is normalised to have elements with the length value 1 to reduce the computational complexity.

1. Sparsity: For the sparsity of the factor  $\mathbf{W}$ , we define the measure as

$$S(\mathbf{W}) = \frac{\sum_{i_2=1}^m \sum_{i_1=1}^k |w_{i_1 i_2}|}{k} \quad (6.4)$$

where  $i_1 = 1, 2, \dots, k$  and  $i_2 = 1, 2, \dots, m$ .  $w_{i_1 i_2}$  is the element value in  $i_2^{th}$  row and  $i_1^{th}$  column. The sparsity equation here is equal to the average of the sum of all the elements in each basis image  $w_{i_1}$ . Note that the less sparsity lead to the less computation which suggests the better data compression result.

2. Orthogonality: In literature, the orthogonality of  $\mathbf{W}$  is often defined as the sum of the inner products of the basis images in  $\mathbf{W}$ . Here we make the slightly modification on it in order to match the same range as sparsity. The function is as follows.

$$O(\mathbf{W}) = \sqrt{\frac{m \sum_{j_1=1, i_1 \neq j_1}^k w_{i_1}^T w_{j_1}}{k(k-1)/2}} \quad (6.5)$$

where  $w_{i_1}$  and  $w_{j_1}$  are both the basis images in  $i_1^{th}$  and  $j_1^{th}$  column of  $\mathbf{W}$  respectively. With the less orthogonality value, the basis images in  $\mathbf{W}$  would be more spatially localized. The sparsity and orthogonality in  $\mathbf{W}$  are closely related. That is, the orthogonality here further enhances the sparsity of  $\mathbf{W}$ .

Combination: This is the objective function which combines both sparsity and orthogonality and the equation is as follows.

$$C(\mathbf{W}) = \frac{S(\mathbf{W}) + O(\mathbf{W})}{2} \quad (6.6)$$

It is equal to the average of the sparsity and orthogonality, so the less  $C(\mathbf{W})$  value lead to the better data compression performance.

## 6.3 Experimental results and analysis

### 6.3.1 Datasets and preprocessing

Two image datasets were used in this chapter which are two common face image datasets called Yale and ORL. The details of the datasets are described below.

**Yale:** This dataset contains 165 grayscale images with  $32 \times 32$  for each image. There are totally 15 subjects and 11 images per subject which include center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink.

**ORL:** This dataset contains 400 images with  $32 \times 32$  for each image. There are totally 40 distinct subjects and 10 images per subject which were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position.

In all the experiments here, images are preprocessed so that all the elements (pixel values) inside the dataset are less than 1. It is achieved by dividing the maximum value in this dataset for each element.

### 6.3.2 Experiment process

As shown in the figure 6.3, we first apply four standard initialization methods to a given dataset and then collect the four different results of pairs of  $(\mathbf{W}, \mathbf{H})$  together. These four pairs of  $(\mathbf{W}, \mathbf{H})$  are inputed into the proposed evolutionary optimization strategy system simultaneously with the different objective function saying sparsity, orthogonality and their combination. This process was run 5 times in order to reduce the influence of the randomness. The dimension of both the two image datasets is reduced to be 25 here. That is, there are totally 25 basis images in each  $\mathbf{W}$ .



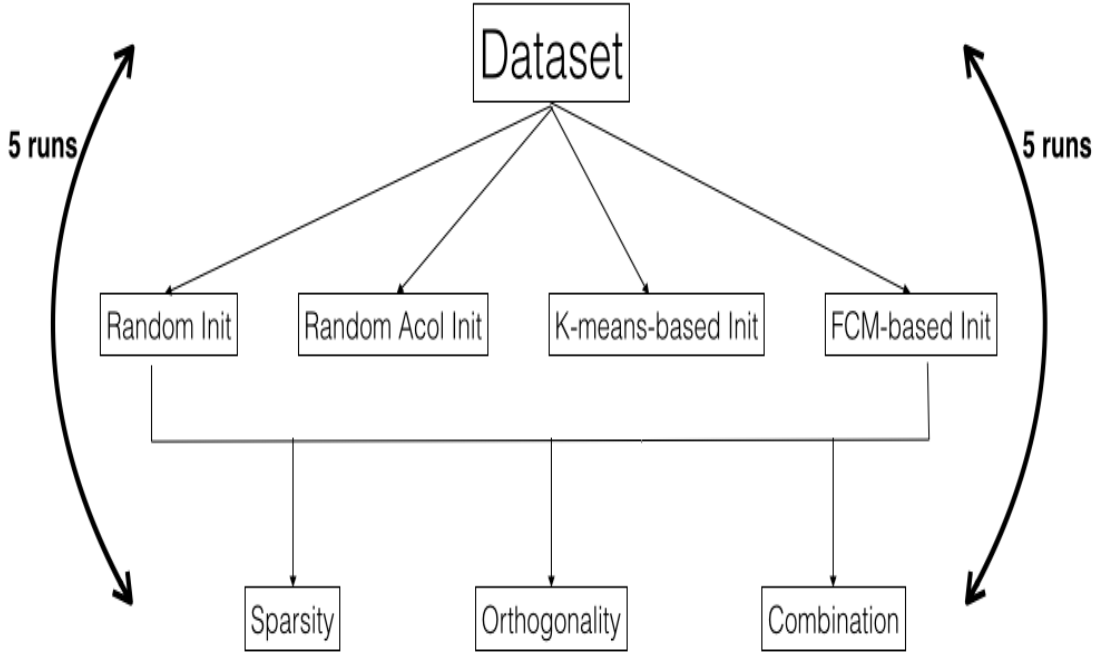


Figure 6.3: The structure of the experimental process

### 6.3.3 Yale image dataset

Figure 6.4 to 6.9 as well as table 6.2 are some results for Yale dataset with the different objective functions. We tested this dataset under  $minterror = 0.3$  and  $iterations = 500$ .  $minterror$  was decided with the one decimal place. Figure 6.4-6.6 shows the performance of sparsity, orthogonality and objective value of the proposed method gets the faster convergence and better data compression performance than the other standard initialized NMF methods during the iterations. In figure 6.4, we only choose the sparsity as the objective function shown in section 6.2.4, so the objective values recorded here is equal to the sparsity values. As shown in the error subplot, the error of the proposed method is always less than 0.3 which satisfies the user's demand of  $minterror = 0.3$ .

Figure 6.7 to 6.9 shows the basis images obtained from the different methods for Yale with the different objective functions. There are totally 25 basis images for each subplot. The rows represent the different methods saying random initialized NMF, random acol initialized NMF, kmeans-based initialized NMF, FCM-based initialized NMF and the proposed evolutionary optimization method from top to bottom. The columns represent the increasing iterations saying 1, 50, 100, 200 and 500 from left to right. We can see from these figures, the proposed method has already gotten the lower sparsity and orthogonality than the other methods after 50 iterations and keeps the stable after about 200 iterations. There are no dramatically change of the basis image for

kmeans-based initialized NMF with the increasing iterations. It gets the raw structure of 25 basis images for random acol initialized NMF, kmeans-based initialized NMF, FCM-based initialized NMF and the proposed evolutionary optimization method at  $iteration = 1$ . However for random initialized NMF, there are too much noises there in its basis images at the beginning of the iterations. The main reason is that the random initialization has nothing to do with the initial values of  $\mathbf{W}$  and  $\mathbf{H}$  while the other methods already works for data compression with predefined values of  $\mathbf{W}$  and  $\mathbf{H}$ .

We have run the experiment process shown in figure6.3 for 5 times and recorded some measurements saying sparsity, orthogonality, error and RAND index each time which is shown in table 6.2. "proposedS", "proposedO" and "proposedC" represent the proposed method with sparsity objective function, orthogonality objective function and their combination objective function respectively. RI, RAI, CI1 (the same as CI1 in chapter 5)and CI2 (the same as CI3 in chapter 5) represent the random initialisation, random acol initialisation, k-mean based initialisation and FCM-based initialisation respectively. All the values are collected from the final iteration with one decimal place. It is clearly shown that the proposed evolutionary optimization strategy always has the less final sparsity, orthogonality and objective function values than the other methods and also its final error is within the predefined range for each run. Here we calculated the RAND index which measures the cluster quality of the factor  $\mathbf{W}$ . Although the RAND value of the proposed method is slightly less than the others, it is still the meaningful RAND value because of its very low decreasing rate. Table 6.3 illustrated the t-test between NMF and ENMF with the different objective functions saying sparsity, orthogonality and their combination. And it showed that the ENMF and NMF with CI1 has the larger difference according to the higher t-test.

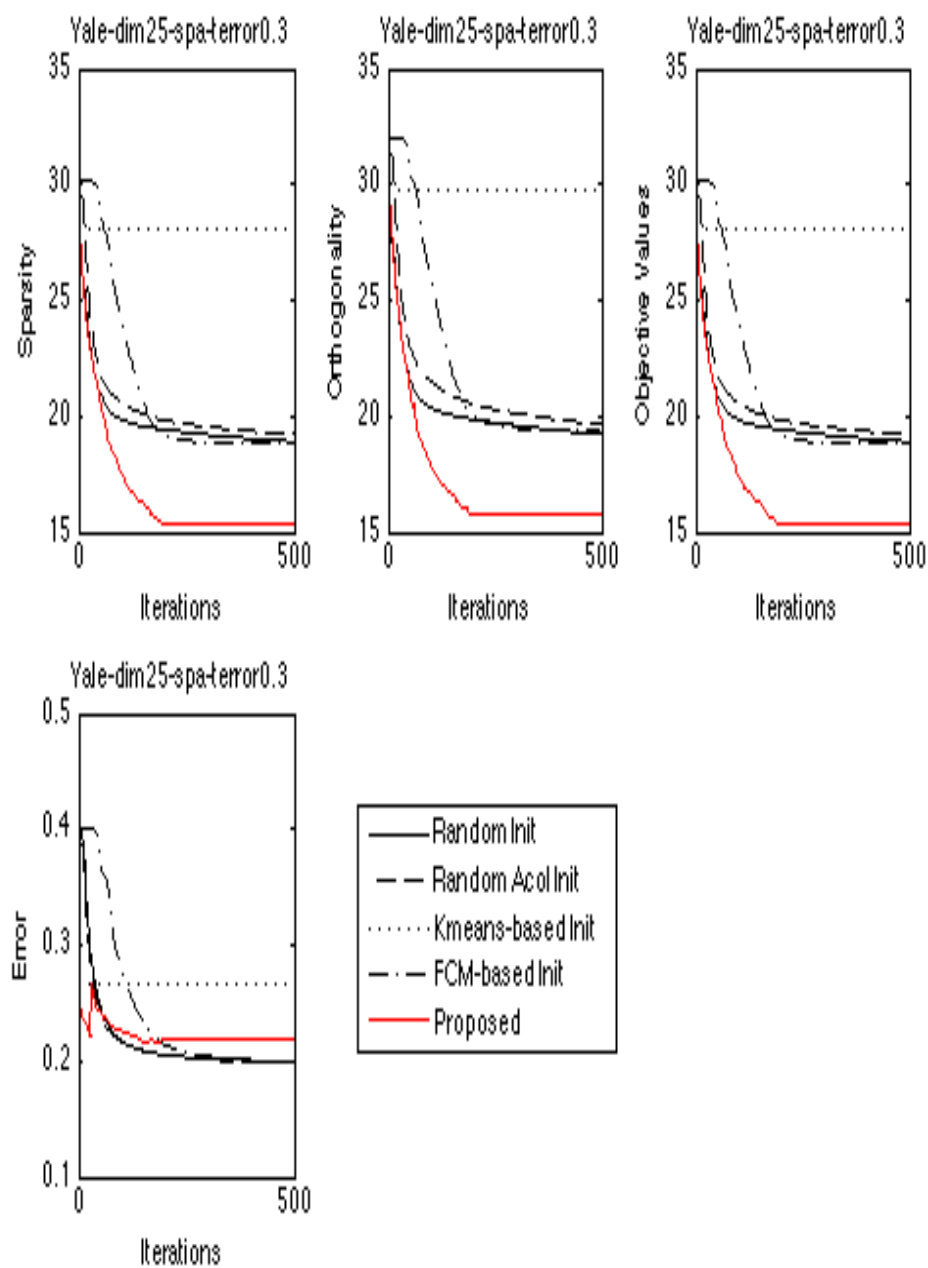


Figure 6.4: The summary results for Yale face dataset under the different data compression methods with the sparsity measure

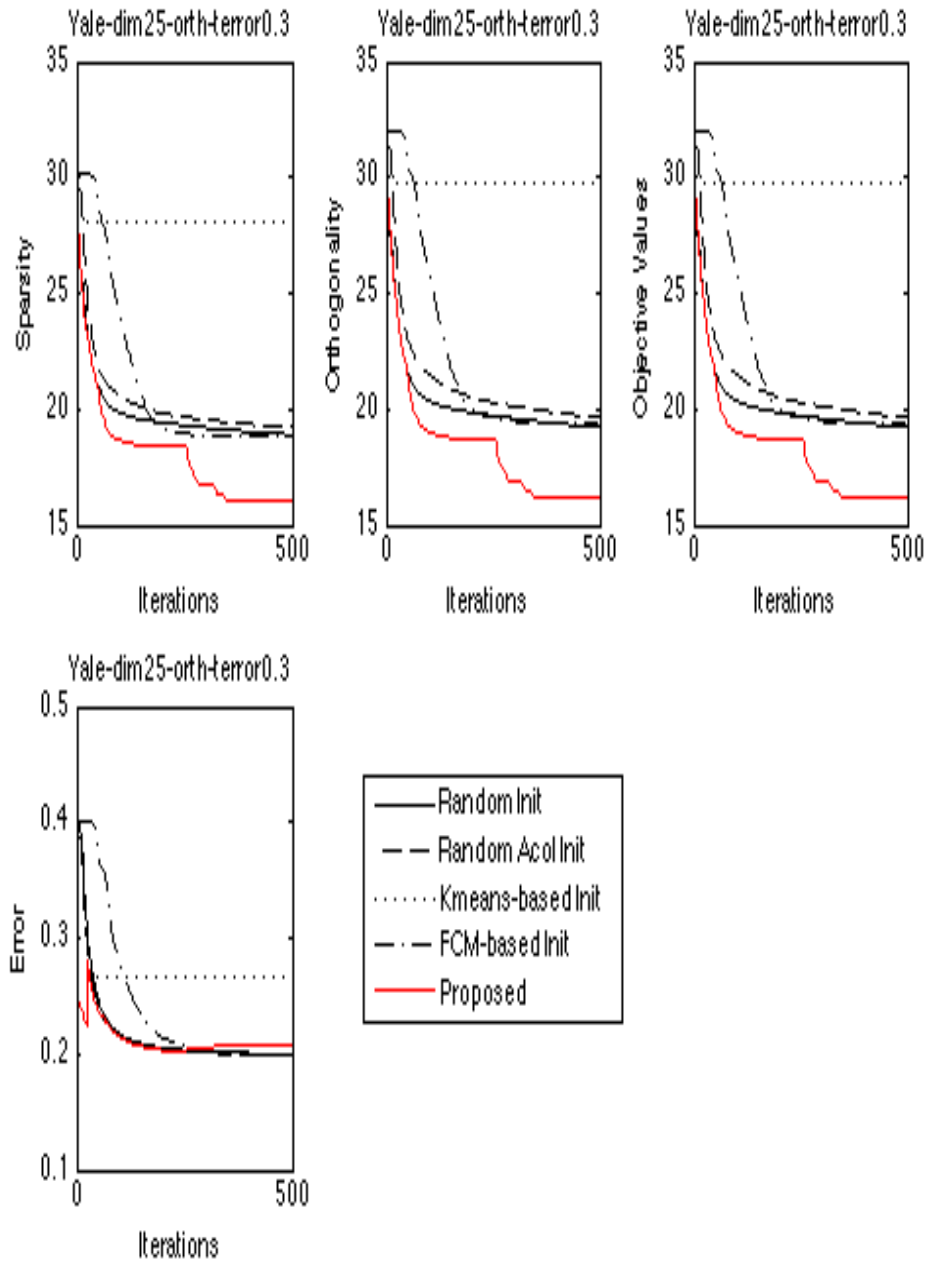


Figure 6.5: The summary results for Yale face dataset under the different data compression methods with the orthogonality measure

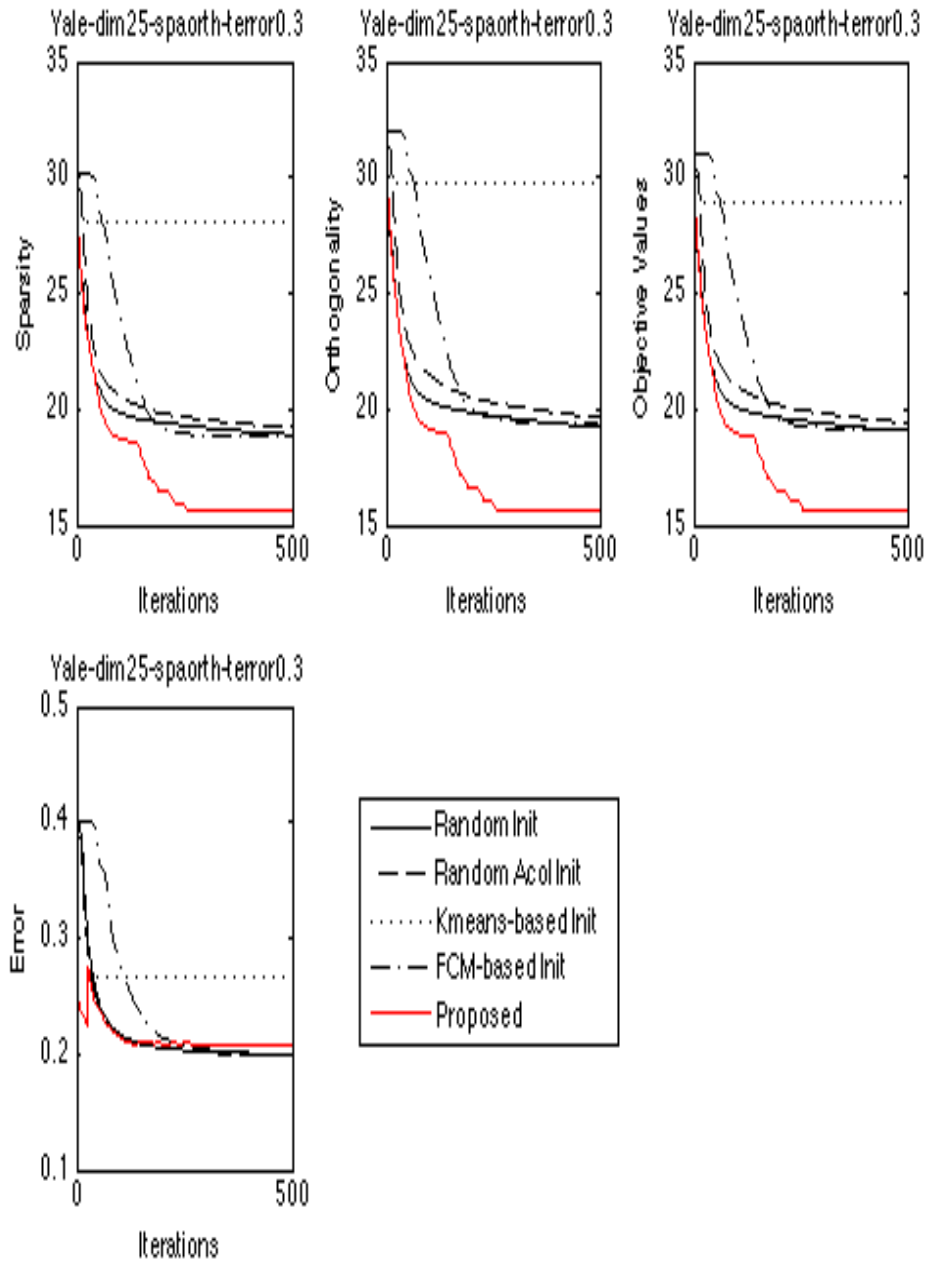


Figure 6.6: The summary results for Yale face dataset under the different data compression methods with both the sparsity and orthogonality measures

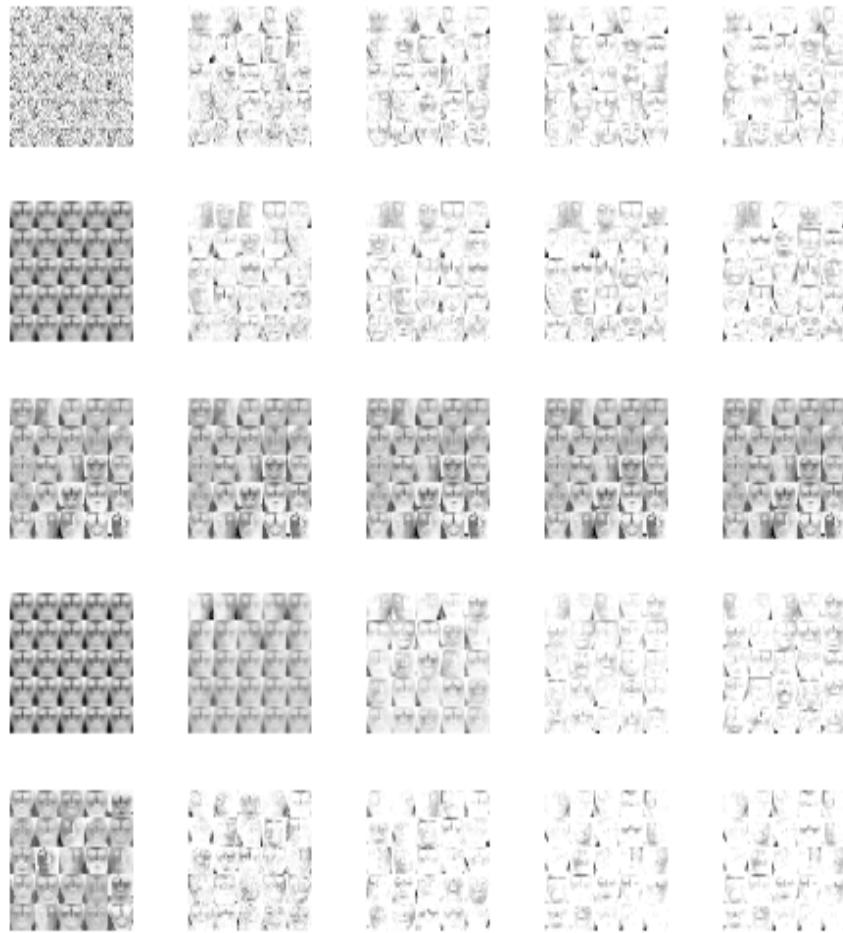


Figure 6.7: The basis images obtained from different methods for Yale with the sparsity measure. (1) 1st row represents the basis images obtained from random initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (2) 2nd row represents the basis images obtained from random acol initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (3) 3rd row represents the basis images obtained from k-means based initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (4) 4th row represents the basis images obtained from FCM-based initialized NMF at the increasing iterations (iter=1, 50, 100, 200, 500) (5) 5th row represents the basis images obtained from the proposed evolutionary optimization strategy at the increasing iterations (iter=1, 50, 100, 200, 500)

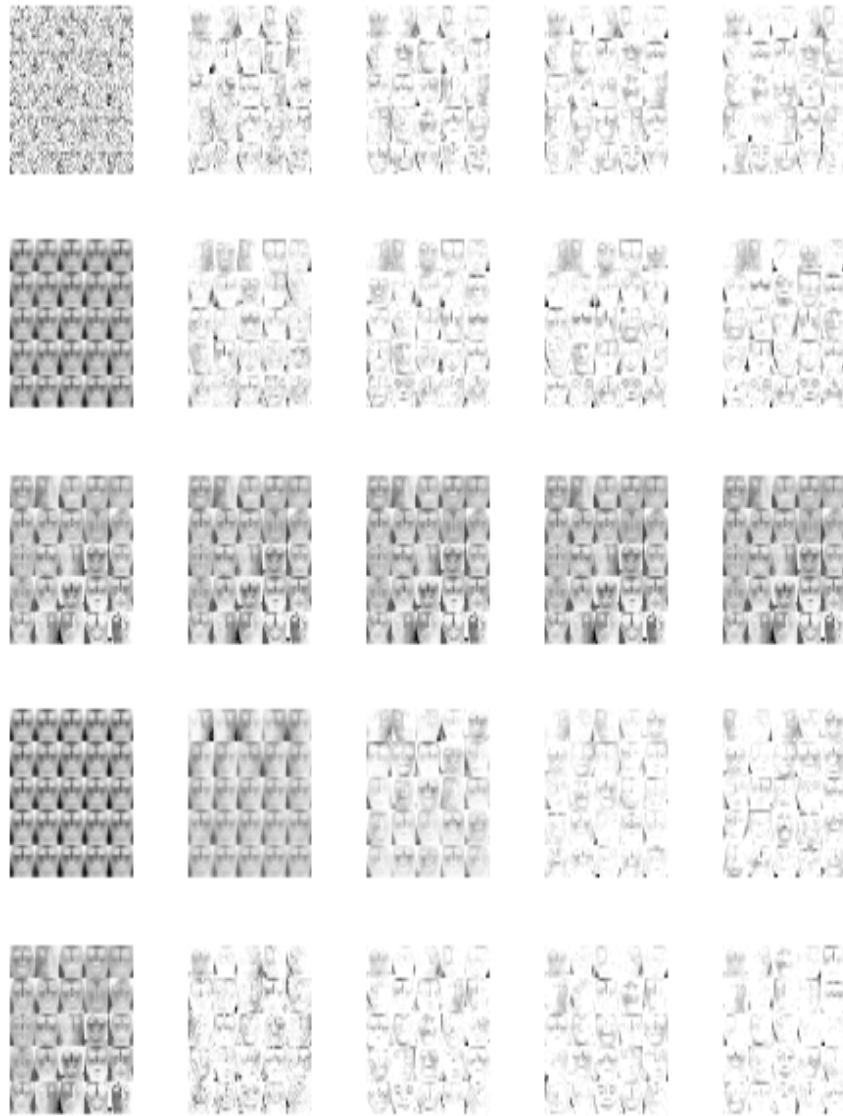


Figure 6.8: The basis images obtained from different methods for Yale with the orthogonality measure which is similar with figure6.7.

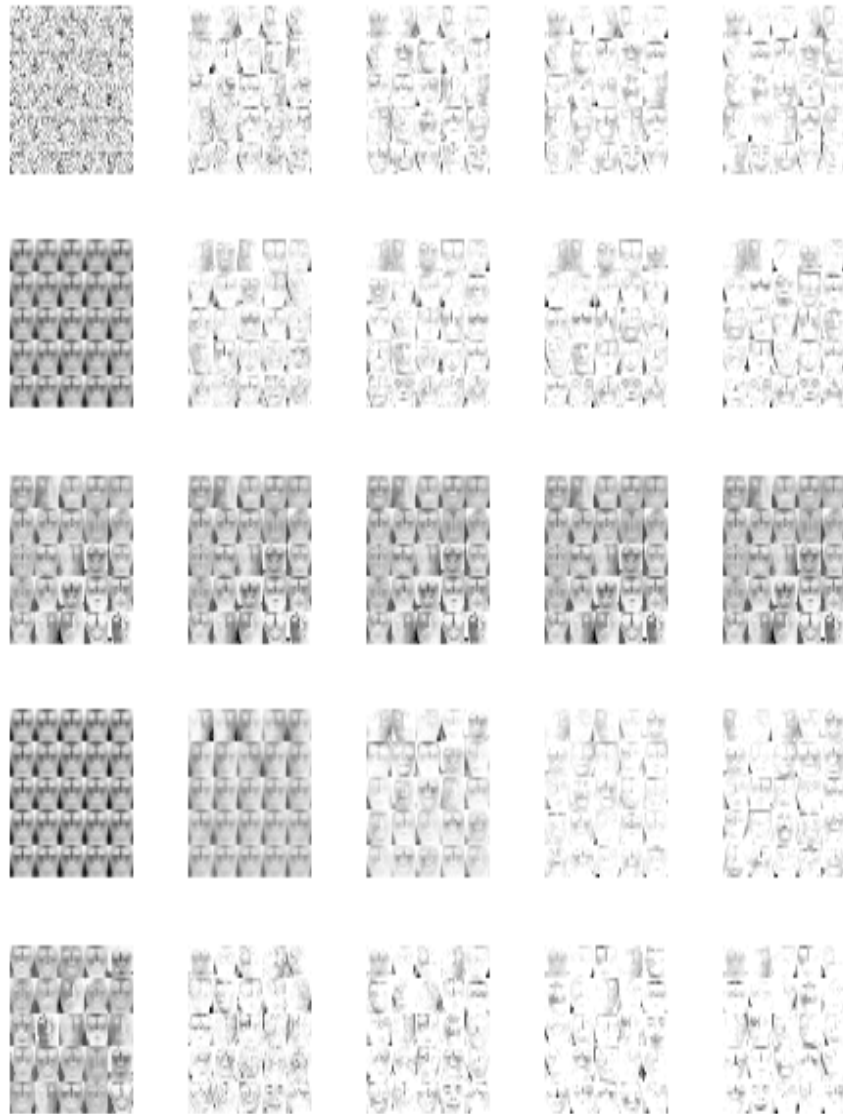


Figure 6.9: The basis images obtained from different methods for Yale with both the sparsity and orthogonality measures which is similar with figure6.7.

### 6.3.4 ORL image dataset

Figure 6.10 to 6.15 as well as table 6.5 are some results for ORL dataset with the different objective functions. Different with Yale's results above, here we tested the *minerror* value to be 0.15 so that the error during the iterations is always less than 0.15 shown in figures 6.10-6.12. The maximum iteration for this dataset is set to be 1000.



The proposed method has already gotten the lower sparsity and orthogonality than the other methods after 100 iterations and keeps the stable after about 500 iterations. In figures 6.13 to 6.15, the columns represent the different iterations saying 1, 100, 200, 500 and 1000 from left to right and the rows represent the different methods as the same as that in figure 6.7. Table 6.4 illustrated the t-test between NMF and ENMF with the different objective functions saying sparsity, orthogonality and their combination. And it showed that the ENMF and NMF with CI1 has the larger difference according to the higher t-test.

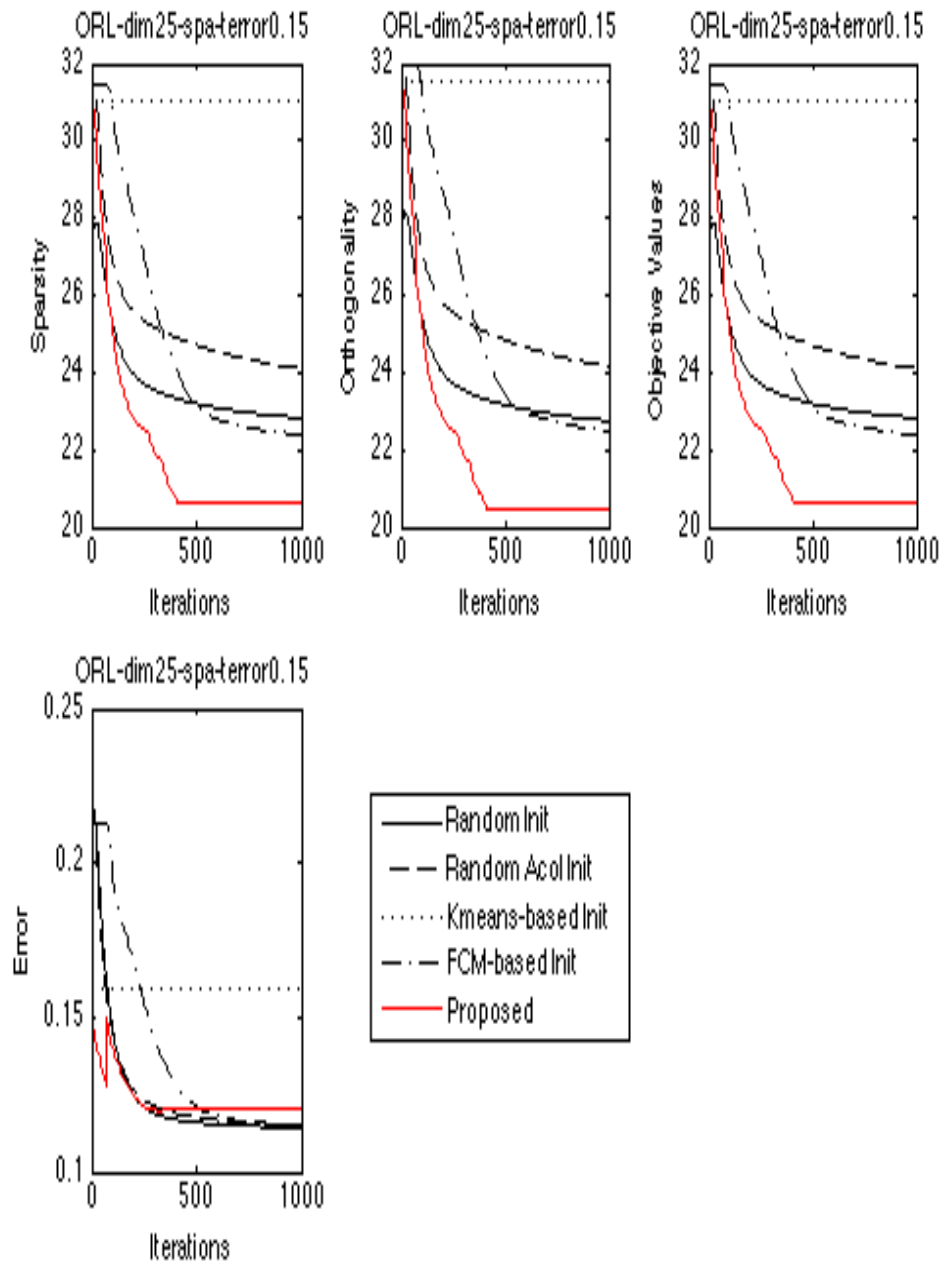


Figure 6.10: The summary results for ORL face dataset under the different data compression methods with the sparsity measure

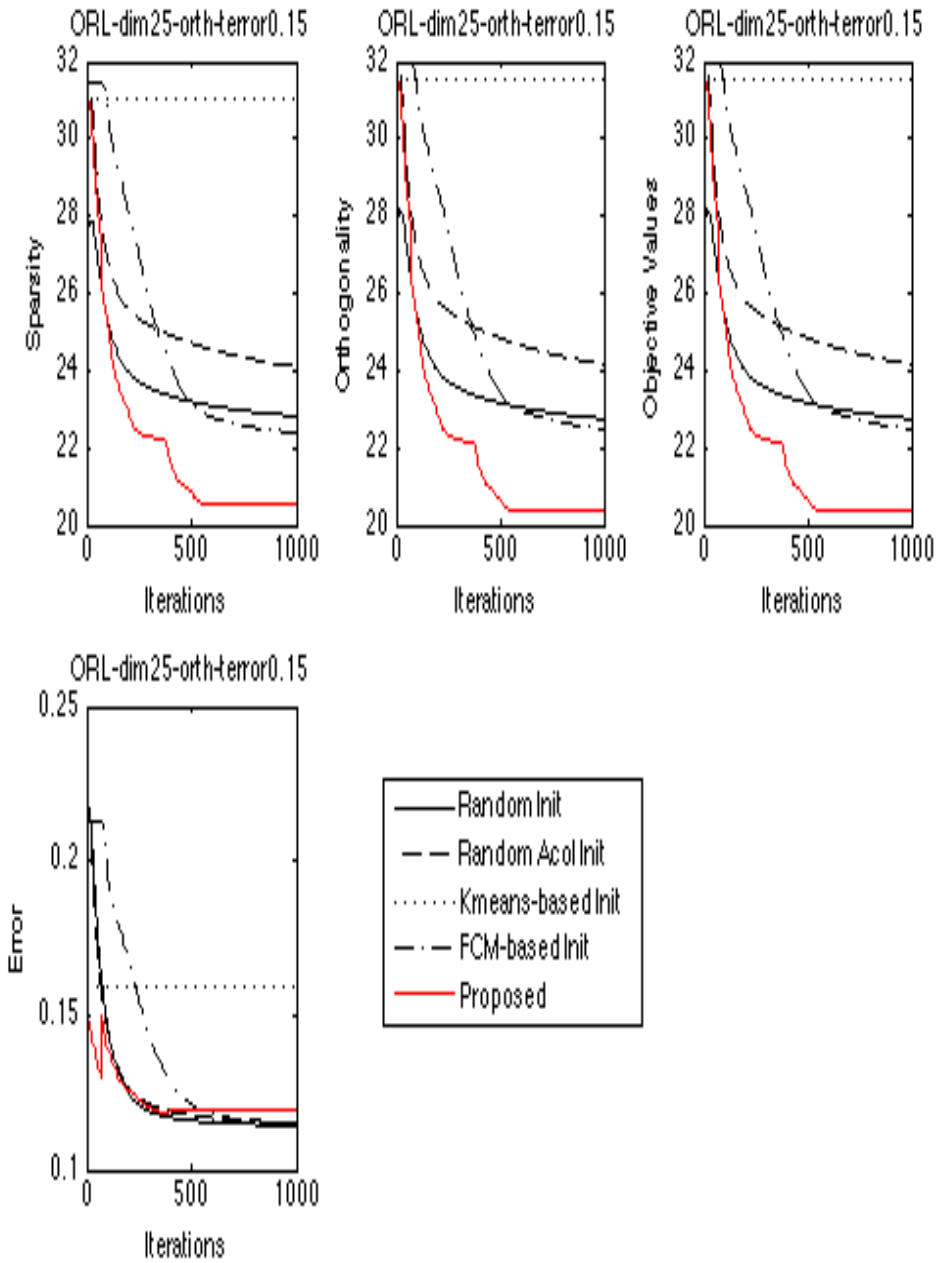


Figure 6.11: The summary results for ORL face dataset under the different data compression methods with the orthogonality measure

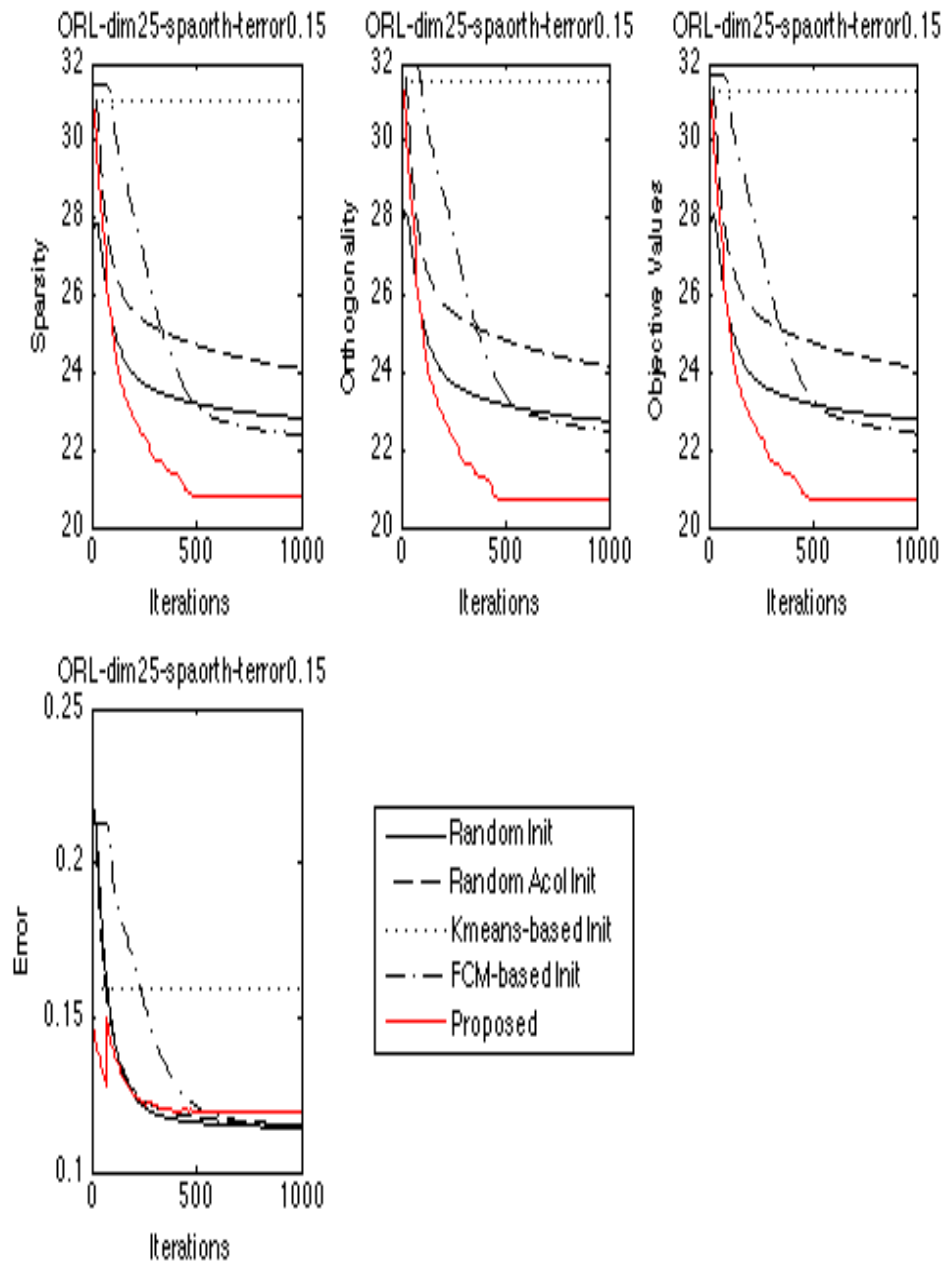


Figure 6.12: The summary results for ORL face dataset under the different data compression methods with both the sparsity and orthogonality measures

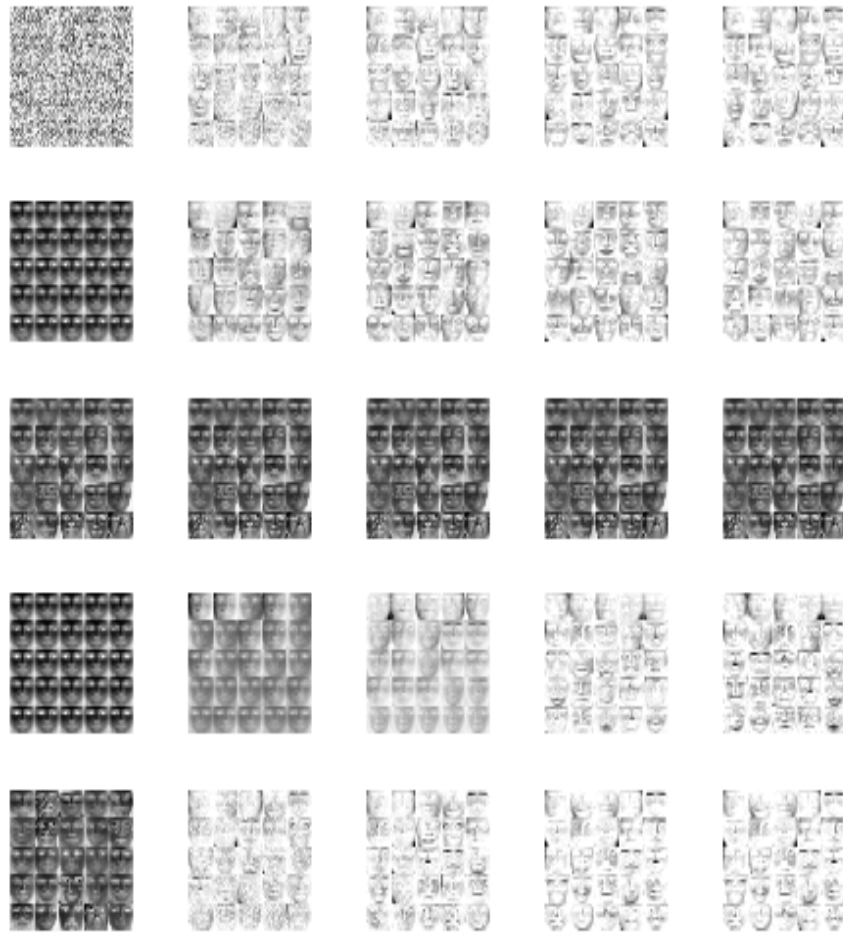


Figure 6.13: The basis images obtained from different methods for ORL with the sparsity measure. (1) 1st row represents the basis images obtained from random initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (2) 2nd row represents the basis images obtained from random acol initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (3) 3rd row represents the basis images obtained from k-means based initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (4) 4th row represents the basis images obtained from FCM-based initialized NMF at the increasing iterations (iter=1, 100, 200, 500, 1000) (5) 5th row represents the basis images obtained from the proposed evolutionary optimization strategy at the increasing iterations (iter=1, 100, 200, 500, 1000)

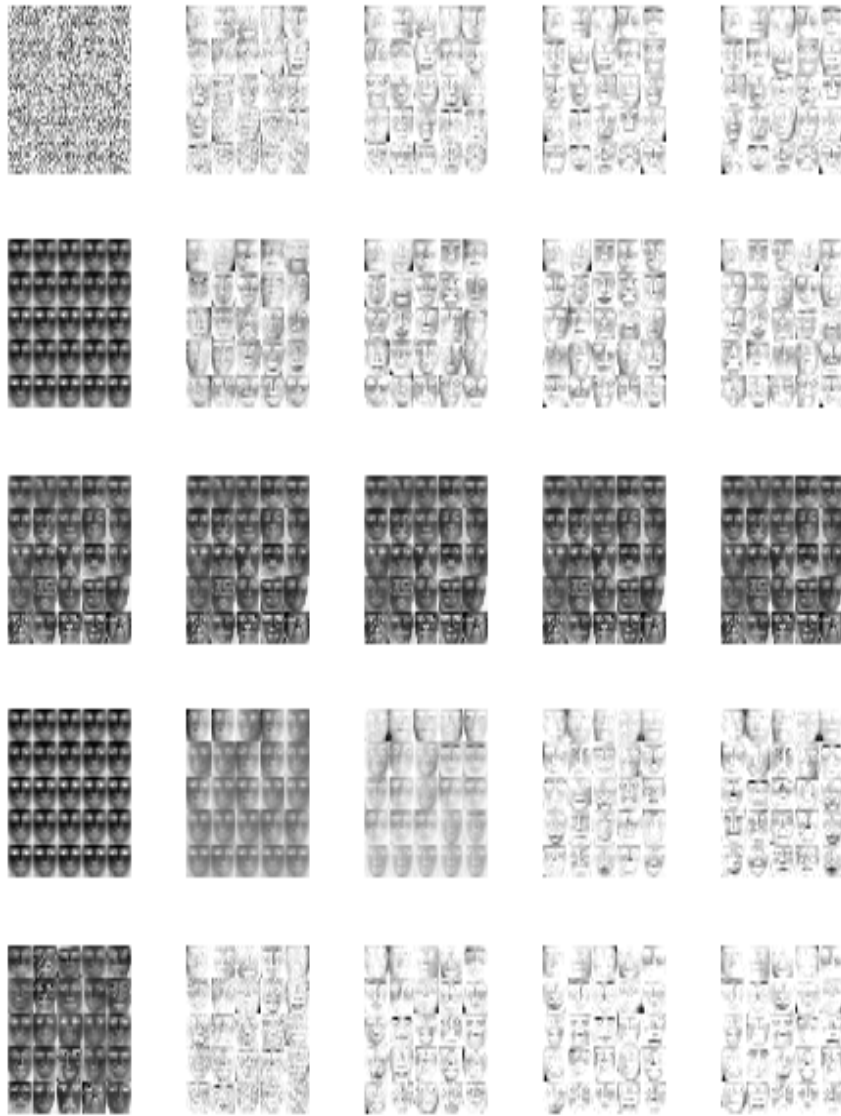


Figure 6.14: The basis images obtained from different methods for ORL with the orthogonality measure which is similar with figure6.13.

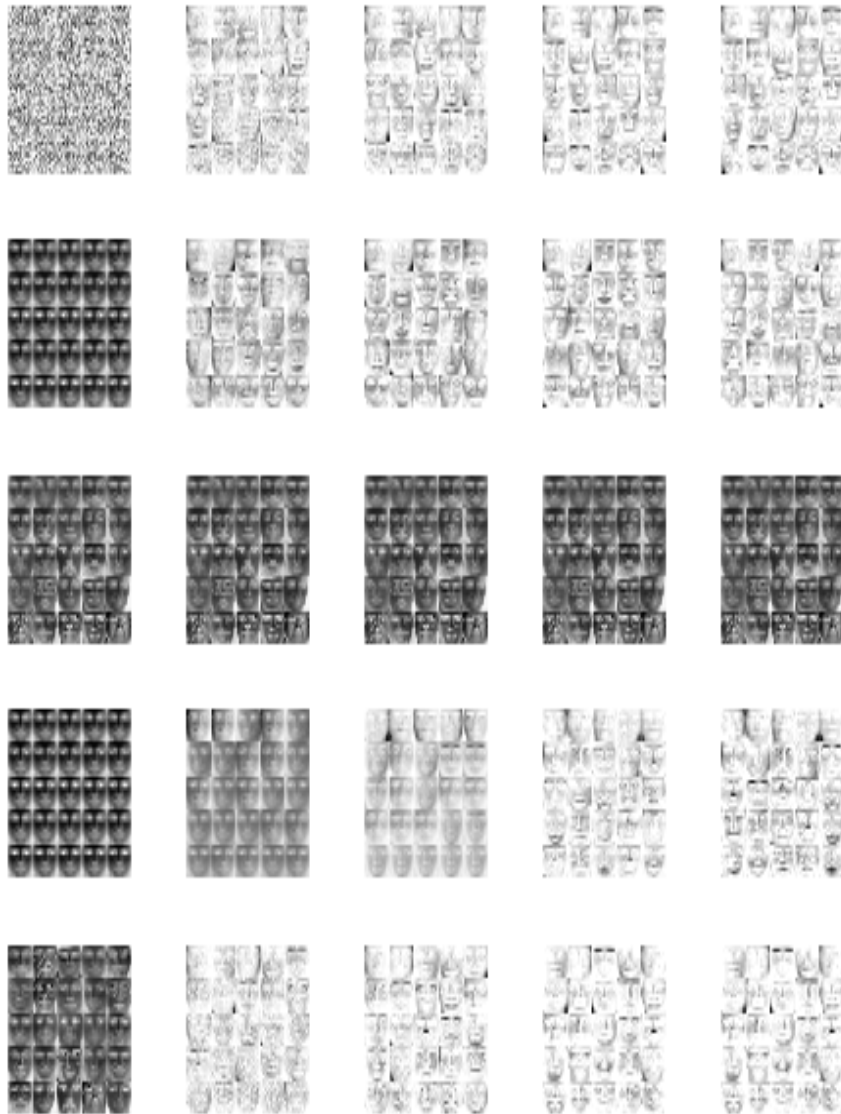


Figure 6.15: The basis images obtained from different methods for ORL with both the sparsity and orthogonality measures which is similar with figure6.13.

## 6.4 Conclusions

In this chapter, we proposed a novel evolutionary optimization strategy for NMF to solve the data compression problem. Four types of the initialization methods are used as the seed knowledge for the proposed method to increase the rate of convergence. The experiments were carried out using two image datasets saying Yale and ORL

and the results show that the proposed evolutionary optimization strategy gets better data compression performance and faster convergence compared with some standard initialized NMF methods. The main contributions of this chapter are listed as follows.

- We proposed a novel evolutionary optimization strategy to solve the different data mining problems by replacing the objective function with a set of measures.
- We proposed three different movements in our system saying original movement, beta movement and best movement to add the new roots of finding the best solution during the iterations.
- The hybrid initialization approach can be used by including the state-of-the-art NMF initialization methods as seed knowledge of our proposed evolutionary optimization strategy and there is no limitation for the number and the type of the initialization methods.
- Our proposed evolutionary optimization strategy is suitable for NMF method and also combines the advantages of some standard optimization methods.
- In Yale and ORL face datasets, our evolutionary optimization strategy outperforms the other standard initialized NMF methods according to a data compression quality measure of sparsity and orthogonality.



Table 6.2: The summary results for Yale face dataset under the different data compression methods with the five measurements saying sparsity, orthogonality, objective value, error and RAND index at the final iteration (iter=500)

Runs	Sparsity	Orthogonality	Error	RAND
1	RI:18.9 RAI:19.2 CI1:28.0 CI2:18.9 proposedS:15.4 proposedO:16.0 proposedC:15.6	RI:19.2 RAI:19.6 CI1:29.7 CI2:19.3 proposedS:15.8 proposedO:16.1 proposedC:15.6	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 proposedS:0.2 proposedO:0.2 proposedC:0.2	RI:91.9 RAI:91.6 CI1:93.2 CI2:91.9 proposedS:89.6 proposedO:91.8 proposedC:92.3
2	RI:19.3 RAI:19.5 CI1:28.3 CI2:18.7 proposedS:14.9 proposedO:15.3 proposedC:15.0	RI:19.7 RAI:20.0 CI1:30.1 CI2:19.1 proposedS:15.6 proposedO:15.0 proposedC:15.2	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 proposedS:0.2 proposedO:0.2 proposedC:0.2	RI:92.4 RAI:91.5 CI1:93.3 CI2:91.6 proposedS:90.7 proposedO:91.1 proposedC:90.6
3	RI:18.7 RAI:19.2 CI1:28.2 CI2:18.4 proposedS:15.3 proposedO:15.9 proposedC:15.6	RI:19.1 RAI:19.6 CI1:29.8 CI2:18.9 proposedS:15.6 proposedO:15.8 proposedC:15.6	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 proposedS:0.2 proposedO:0.2 proposedC:0.2	RI:92.1 RAI:92.0 CI1:93.3 CI2:91.4 proposedS:90.8 proposedO:91.6 proposedC:90.2
4	RI:18.4 RAI:19.5 CI1:28.0 CI2:18.2 proposedS:14.3 proposedO:16.2 proposedC:15.7	RI:18.7 RAI:19.9 CI1:29.5 CI2:18.6 proposedS:14.3 proposedO:16.2 proposedC:15.6	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 proposedS:.2 proposedO:0.2 proposedC:0.2	RI:91.4 RAI:92.2 CI1:92.7 CI2:91.6 proposedS:90.9 proposedO:91.6 proposedC:90.6
5	RI: 18.8 RAI:19.2 CI1:28.2 CI2:18.7 proposedS:16.4 proposedO:15.7 proposedC:14.8	RI:19.3 RAI:19.7 CI1:29.9 CI2:19.0 proposedS:16.5 proposedO:15.8 proposedC:15.1	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 proposedS:0.2 proposedO:0.2 proposedC:0.2	RI:91.8 RAI:91.6 CI1:93.1 CI2:91.6 proposedS:91.5 proposedO:91.1 proposedC:90.5
Average	RI: 18.8 RAI:19.3 CI1:28.1 CI2:18.6 <b>proposedS:15.3</b> <b>proposedO:15.8</b> <b>proposedC:15.3</b>	RI:19.2 RAI:19.8 CI1:29.8 CI2:19.0 <b>proposedS:15.6</b> <b>proposedO:15.8</b> <b>proposedC:15.4</b>	RI:0.2 RAI:0.2 CI1:0.3 CI2:0.2 <b>proposedS:0.2</b> <b>proposedO:0.2</b> <b>proposedC:0.2</b>	RI:91.9 RAI:91.8 CI1:93.1 CI2:91.6 proposedS:90.7 proposedO:91.4 proposedC:90.8

Table 6.3: Statistical significance test (t-test) for the average ENMF and NMF after 5 runs shown in table 6.2

Init.	Sparsity	Orthogonality	Combination
RI	0.07	0.06	0.07
RAI	0.08	0.07	0.08
CI1	0.26	0.25	0.26
CI2	0.07	0.05	0.07

Table 6.4: Statistical significance test (t-test) for the average ENMF and NMF after 5 runs shown in table 6.5

Init.	Sparsity	Orthogonality	Combination
RI	0.05	0.05	0.05
RAI	0.08	0.07	0.07
CI1	0.2	0.19	0.20
CI2	0.05	0.04	0.04

Table 6.5: The summary results for ORL face dataset under the different data compression methods with the five measurements saying sparsity, orthogonality, objective value, error and RAND index at the final iteration (iter=500)

Runs	Sparsity	Orthogonality	Error	RAND
1	RI: 22.8 RAI:24.1 CI1:31.1 CI2:22.4 proposedS:20.6 proposedO:20.5 proposedC:20.8	RI:22.8 RAI:24.2 CI1:31.5 CI2:22.5 proposedS:20.4 proposedO:20.4 proposedC:20.7	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 proposedS:0.1 proposedO:0.1 proposedC:0.1	RI:95.1 RAI:95.0 CI1:95.2 CI2:94.8 proposedS:94.4 proposedO:94.7 proposedC:94.1
2	RI: 23.0 RAI:23.9 CI1:31.0 CI2:22.5 proposedS:20.1 proposedO:20.4 proposedC:20.0	RI:22.9 RAI:24.0 CI1:31.5 CI2:22.6 proposedS:19.9 proposedO:20.3 proposedC:19.8	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 proposedS:0.1 proposedO:0.1 proposedC:0.1	RI:94.9 RAI:95.1 CI1:94.9 CI2:95.0 proposedS:94.0 proposedO:93.6 proposedC:94.0
3	RI: 23.1 RAI:24.1 CI1:31.0 CI2:22.8 proposedS:19.8 proposedO:20.7 proposedC:20.5	RI:23.1 RAI:24.2 CI1:31.5 CI2:22.9 proposedS:19.5 proposedO:20.5 proposedC:20.3	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 proposedS:0.1 proposedO:0.1 proposedC:0.1	RI:95.1 RAI:94.9 CI1:94.8 CI2:94.9 proposedS:94.1 proposedO:95.0 proposedC:94.5
4	RI: 22.8 RAI:24.3 CI1:31.1 CI2:22.4 proposedS:20.9 proposedO:20.7 proposedC:20.9	RI:22.8 RAI:24.4 CI1:31.6 CI2:22.6 proposedS:20.8 proposedO:20.6 proposedC:20.8	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 proposedS:0.1 proposedO:0.1 proposedC:0.1	RI:94.9 RAI:95.3 CI1:95.4 CI2:94.8 proposedS:94.1 proposedO:94.4 proposedC:94.2
5	RI: 22.8 RAI:24.1 CI1:31.0 CI2:22.4 proposedS:19.9 proposedO:20.8 proposedC:20.6	RI:22.8 RAI:24.2 CI1:31.5 CI2:22.5 proposedS:19.7 proposedO:20.6 proposedC:20.4	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 proposedS:0.1 proposedO:0.1 proposedC:0.1	RI:94.9 RAI:95.0 CI1:95.0 CI2:94.9 proposedS:94.3 proposedO:94.5 proposedC:93.7
Average	RI: 22.9 RAI:24.1 CI1:31.0 CI2:22.5 <b>proposedS:20.3</b> <b>proposedO:20.6</b> <b>proposedC:20.6</b>	RI:22.9 RAI:24.2 CI1:31.5 CI2:22.6 <b>proposedS:20.1</b> <b>proposedO:20.5</b> <b>proposedC:20.4</b>	RI:0.1 RAI:0.1 CI1:0.2 CI2:0.1 <b>proposedS:0.1</b> <b>proposedO:0.1</b> <b>proposedC:0.1</b>	RI:95.0 RAI:95.1 CI1:95.1 CI2:94.9 proposedS:94.2 proposedO:94.4 proposedC:94.1

## Chapter 7

# Application of EEG Dataset

This chapter is the report for the work in [20] which is about linking brain responses to naturalistic music through analysis of ongoing EEG and stimulus features. Our work in this chapter is the clustering analysis of EEG dataset. Three clustering algorithms—k-means, k-medoids, and hierarchical clustering as well as ensemble clustering were applied for this work. Work of this chapter is published in:

FY Cong, V Alluri, AK Nandi, P Toivainen, Rui Fa, Basel Abu-Jamous, Liyun Gong, BGW Craenen, H Poikonen, M Huotilainen, T Ristaniemi, Linking Brain Responses to Naturalistic Music through Analysis of Ongoing EEG and Stimulus Features, *IEEE Transactions on Multimedia*, 15(5):1060-1069, 2013. (IF=1.776)

### 7.1 Introduction

In this chapter, we provide the clustering analysis to the preprocessed EEG data. The EEG signal was collected using 64 electrodes from 14 individual subjects. Afterwards, the signal from each electrode and each subject was processed and 64 independent components (ICs) were extracted. Thus in this case, the data matrix is in the form of 896 rows by 64 columns, where each row represents each IC of each subject and each column represents each electrode. The rest of this chapter is organized as follows: Sec. 7.2 states our clustering algorithms of EEG data and the results are given in Sec. 7.3. Finally, conclusions are made in Sec. 7.4.

### 7.2 Clustering methods

Clustering algorithms play a central part in the analysis of EEG data. There are numbers of clustering algorithms in the literature [113, 51, 28, 27, 69, 45, 31, 70, 44]. We investigated many clustering algorithms in the EEG data. Considering the effectiveness of the clustering algorithms, here we present five clustering algorithms and their results here, namely k-means [28, 27], k-medoids [69, 70], hierarchical clustering (HC) [45, 31] with ward linkage, HC with complete linkage and HC with average linkage. The gen-

eral clustering algorithms were reviewed in chapter 2. We also applied the ensemble clustering in this work as follows.

### **Ensemble Clustering**

By scrutinising the results generated by each of the individual clustering methods, ensemble clustering generates a consensus clustering result which collectively summarises all of the individual results [53, 83, 104]. The method we used, consensus fuzzy partition matrix binarisation (CFPMB) [4] starts by fusing the individual partitions (clustering results) into a single fuzzy consensus partition matrix (CFPM) which is then binarised to generate a consensus binary partition matrix (CBPM). This binarisation step considers how often any group of elements were included in the same cluster by each of the individual results. Binarisation can be tuned to result in tight clusters that only include the elements that are included in the same cluster by all or most of the individual clustering methods. This means that at these tightly tuned cases, many clusters will be empty and many elements will not be included in any of the clusters. The method can also be tuned to generate wide overlapping clusters that assign any doubtful element to all of its possible clusters. At this stage of this work, tuning the binarisation step to generate tight clusters seems more beneficial such that only pure clusters' cores are generated and the research can then be focused. Two of the proposed binarisation techniques are of interest in this research currently; value thresholding binarisation (VTB) and difference thresholding binarisation (DTB). Each of these techniques can be tuned through a parameter that ranges from zero to one. When the parameter is zero, VTB produces the absolute widest possible clusters and then as the parameter is increased, the clustered are tightened up to the absolute tightest clusters at the value of one. DTB generates complementary clusters when the parameter is zero and tightens them as the parameter is increased but in a different way than VTB. When the parameter of DTB reaches one then it is the absolute tightest case and is equivalent to VTB with a unity parameter. To get tighter clusters than the conventional complementary case, we use VTB with parameter values larger than 0.5 and DTB with parameter values larger than zero.

## **7.3 Experimental Results**

### **7.3.1 EEG data structure**

The dataset called EEG data has been made available to us. It represents essentially the result of Fengyu Cong's independent component analysis (ICA) on the 'raw' EEG. It is stored in a data format suitable to be loaded into Matlab as a matrix, in the case named X. Below I explain in some detail how the data is structured, it can be described as being an  $N \times M$  matrix with  $N$  the number of rows and  $M$  the number of columns. The  $N \times M = 896 \times 64$  matrix for the EEG dataset is structured as follows:

1. Each row represents the ICA representation of a sequence of electrode values, with each cell along the row representing a single electrode. There are  $M = 64$  electrodes/columns.
2. The order for the rows is as follows: there were 14 participants in the experiment, and 64 independent components were selected/generated. For each independent component (row) all participants are listed first. As such, working down the rows we first get the first independent component for all 14 participants, then the second independent component for all 14 participants, etc. In all there are  $14 \times 64 = 896$  rows in the matrix

More explicit: Where  $IC_j^i = \{v_1, v_2, \dots, v_{64}\}$  is the  $j = 1, 2, \dots, 64$  independent component for participant  $i = 1, 2, \dots, 14$  with  $\{v_1, v_2, \dots, v_{64}\}$  the independent component value vector, the independent component matrix  $ICM^{EEG}$  is then:

$$ICM^{EEG} = \begin{pmatrix} IC_1^1 \\ IC_1^2 \\ \vdots \\ IC_1^{14} \\ IC_2^1 \\ IC_2^2 \\ \vdots \\ IC_2^{14} \\ \vdots \\ IC_{64}^1 \\ IC_{64}^2 \\ \vdots \\ IC_{64}^{14} \end{pmatrix} = \begin{pmatrix} v_1 & v_2 & \dots & v_{64} \\ v_1 & v_2 & \dots & v_{64} \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_{64} \\ v_1 & v_2 & \dots & v_{64} \\ v_1 & v_2 & \dots & v_{64} \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_{64} \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_{64} \\ v_1 & v_2 & \dots & v_{64} \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_{64} \end{pmatrix}$$

Figure 7.1: The independent component matrix

### 7.3.2 Results

#### K-Means with random initialization to the raw data

This is a preliminary experiment to help us to know more about the data. We look for  $K = 64$  clusters in the dataset. The histogram of the clustering result, which shows the number of memberships for each cluster, is shown in Figure 7.2. The results show that the difference between the largest and the smallest clusters is dramatic, namely 83 vs. 1. We show all profiles of the members in the largest cluster, (No. 33 cluster), in Figure 7.3. Surprisingly, the patterns in this cluster are quite different and they should

not really belong to one cluster. We also show one of single-member clusters, (No. 31 cluster) in Figure 7.4.

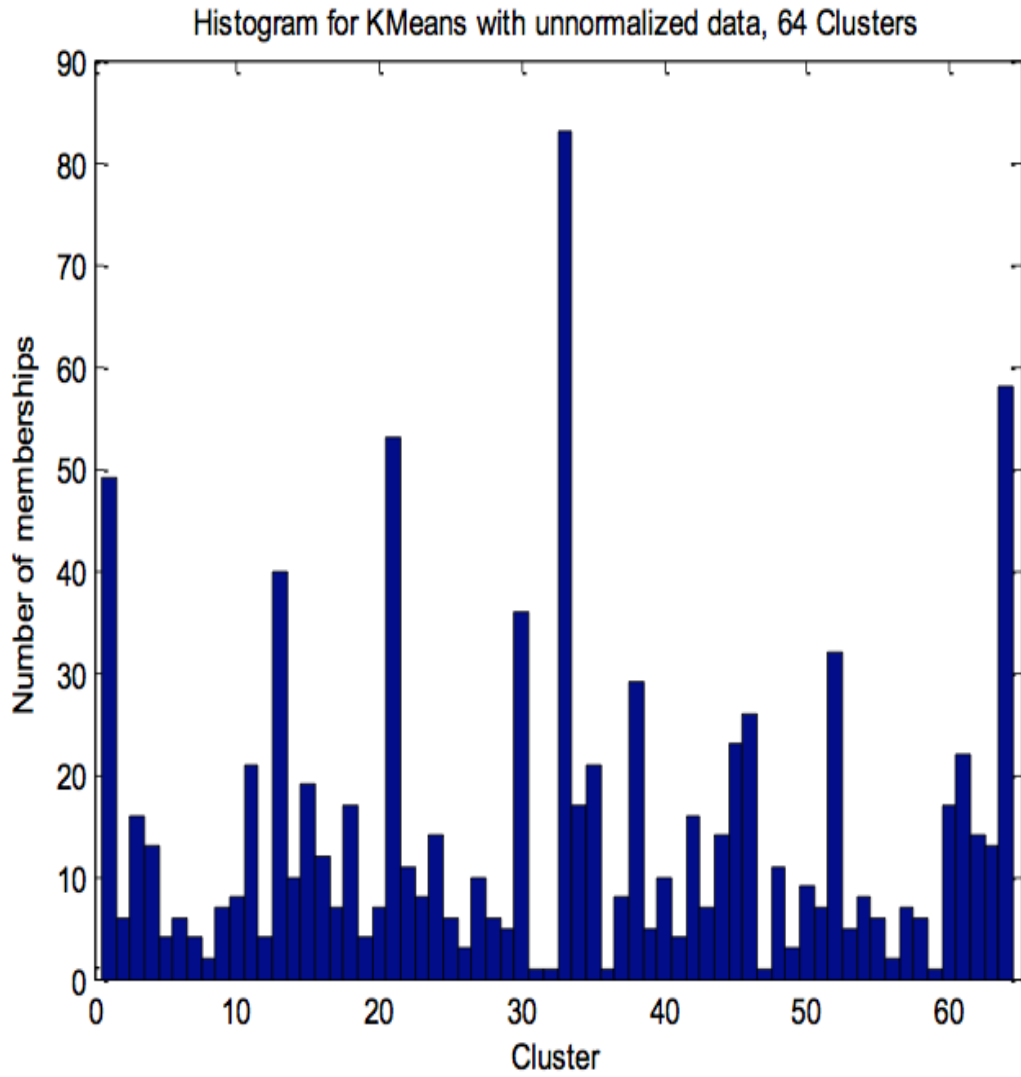


Figure 7.2: Histogram of k-means clustering result with  $K = 64$  from the unnormalized data

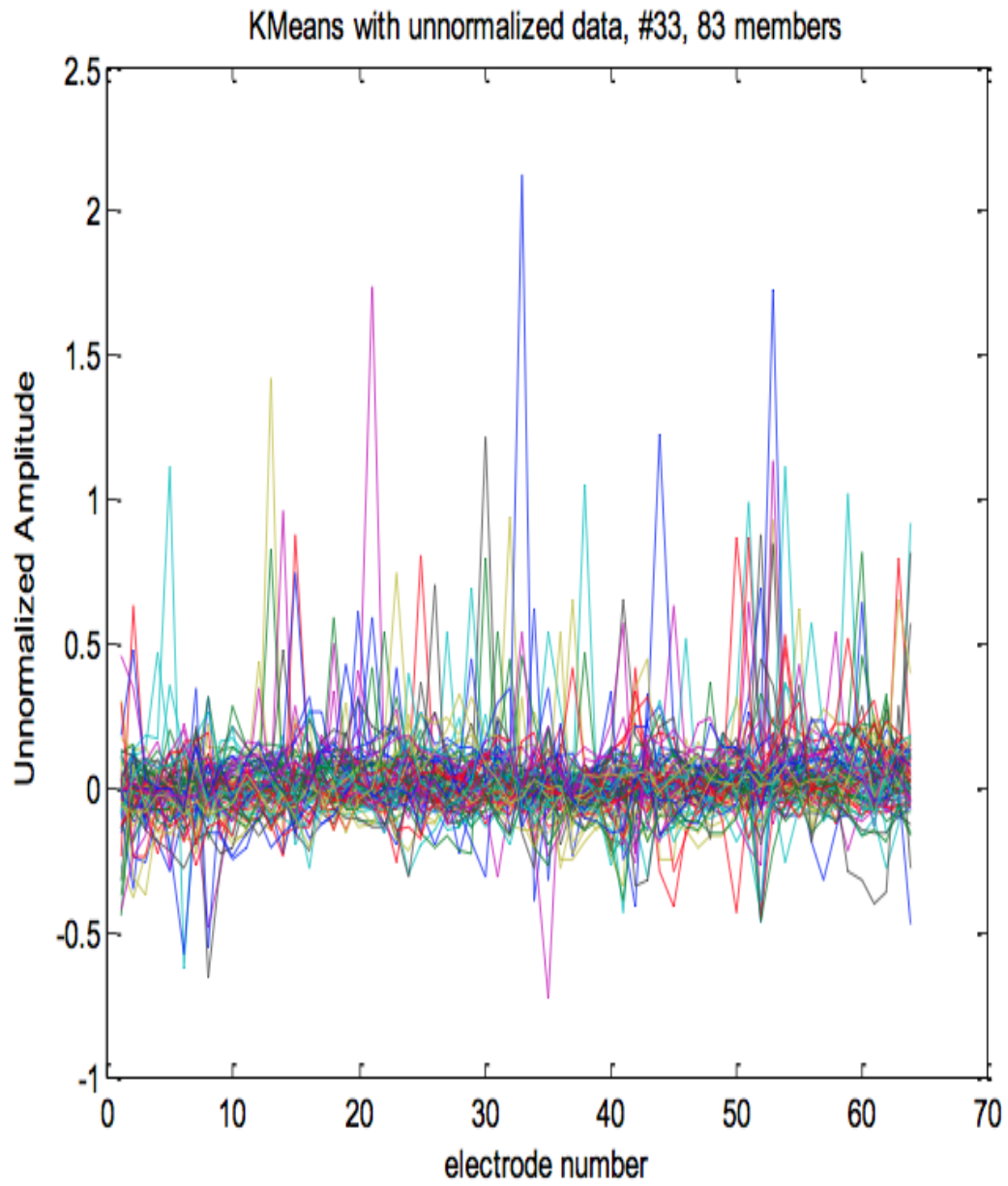


Figure 7.3: The largest cluster (No 33 cluster) in the k-means clustering result with  $K = 64$  from the unnormalized data



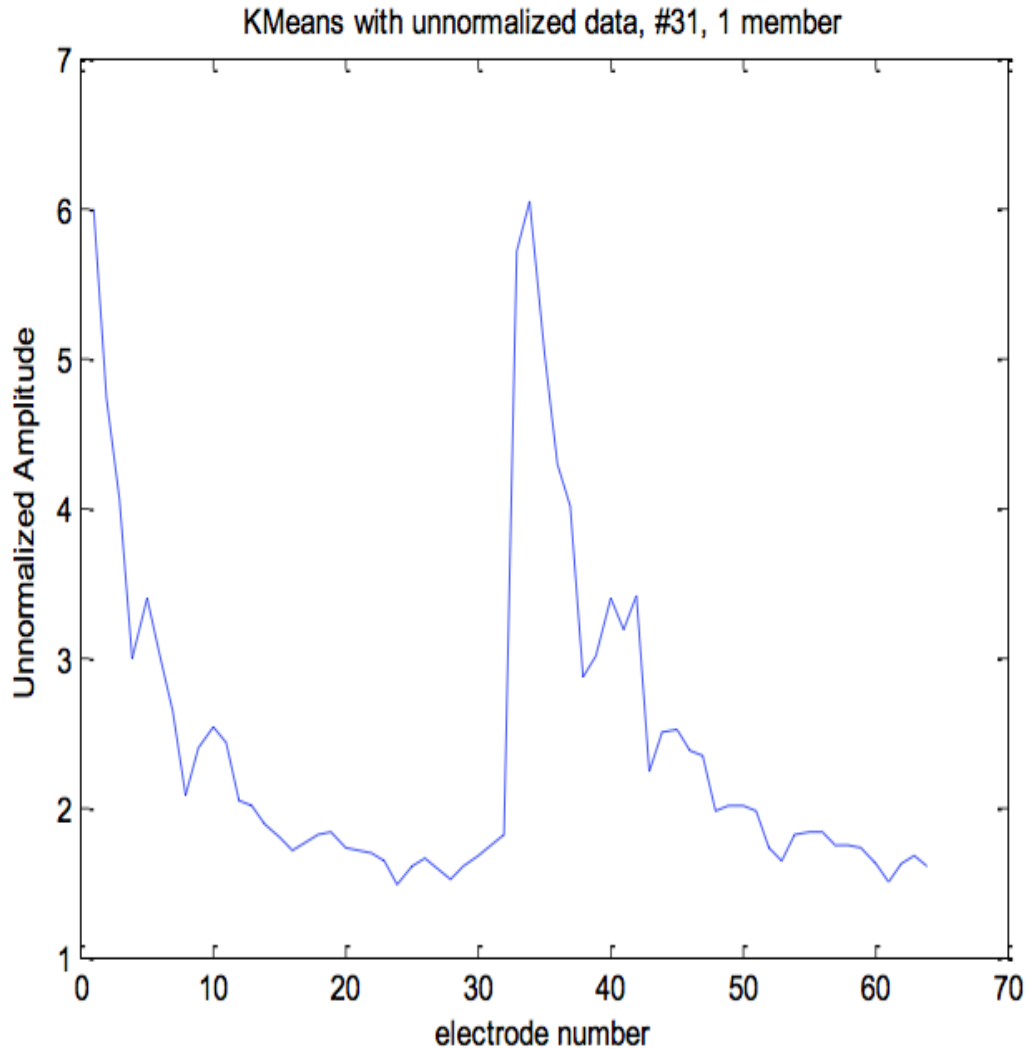


Figure 7.4: One of single-member clusters, No 31 cluster, in the k-means clustering result with  $K = 64$  from the unnormalized data

#### **K-Means with deterministic initialization to the normalized data**

In order to achieve stable and reasonable clustering results, we make two changes to the above experiment. Firstly, we employ deterministic initialization algorithm to avoid providing different clustering results for every experiment. Kaufman approach (KA) [70] (shown in section 2.2.2), which was reported to be superior to other initialization approaches [44], is employed in this work. Secondly, we normalize (standardize) the data to make each row zero mean and unit standard deviation. Now we also look for  $K = 64$  clusters and find that the results are much different with the previous ones. The histogram of the clustering result is shown in Figure 7.5. It is worth noting that all the numbers of memberships in this case are more similar. We find the cluster containing the member in single-member cluster in the last experiment, which was

shown in Figure 7.4 and plot its memberships in Figure 7.6. The patterns of the same cluster in unnormalized data are shown in Figure 7.7.

It reflects that some objects with similar patterns are grouped into different clusters when clustering the unnormalized data.

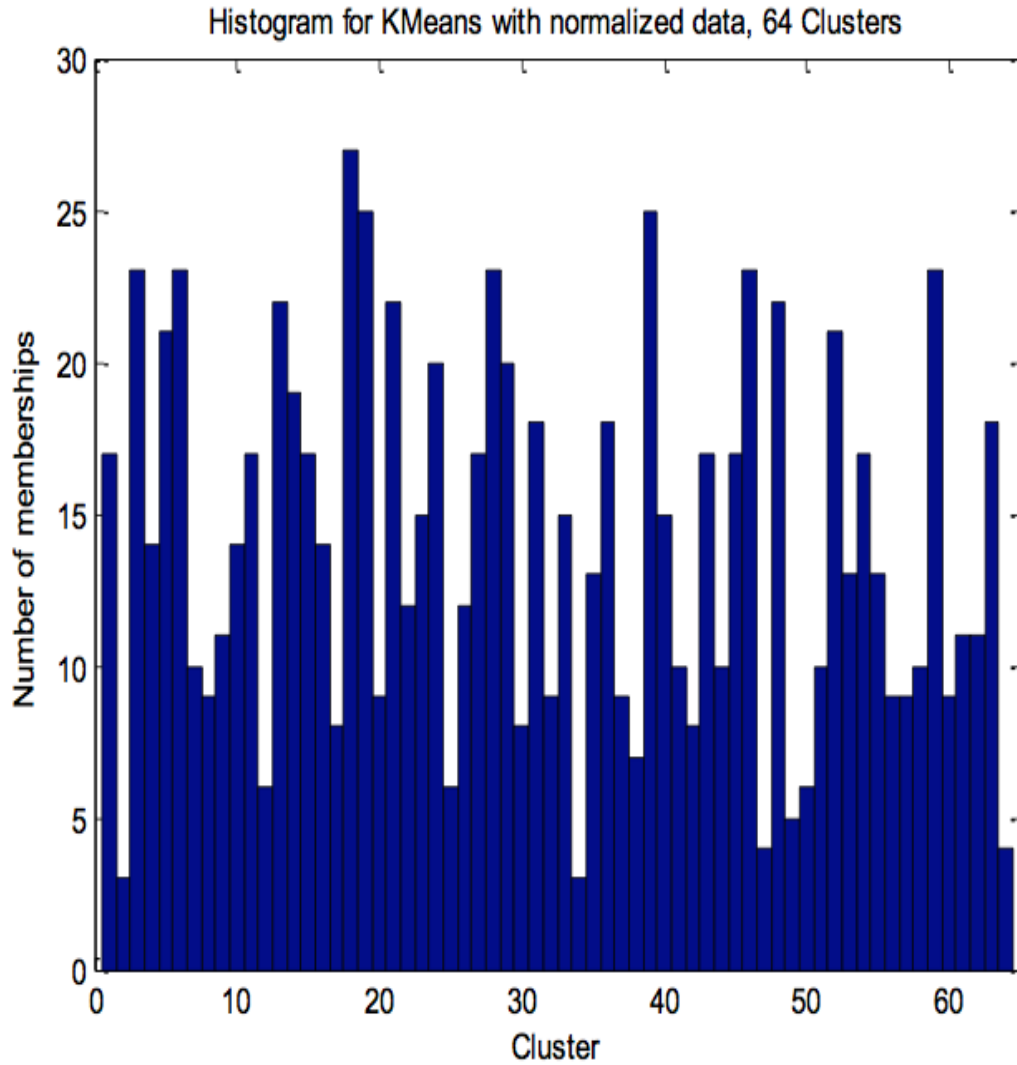


Figure 7.5: Histogram of k-means clustering result with  $K = 64$  from the normalized data

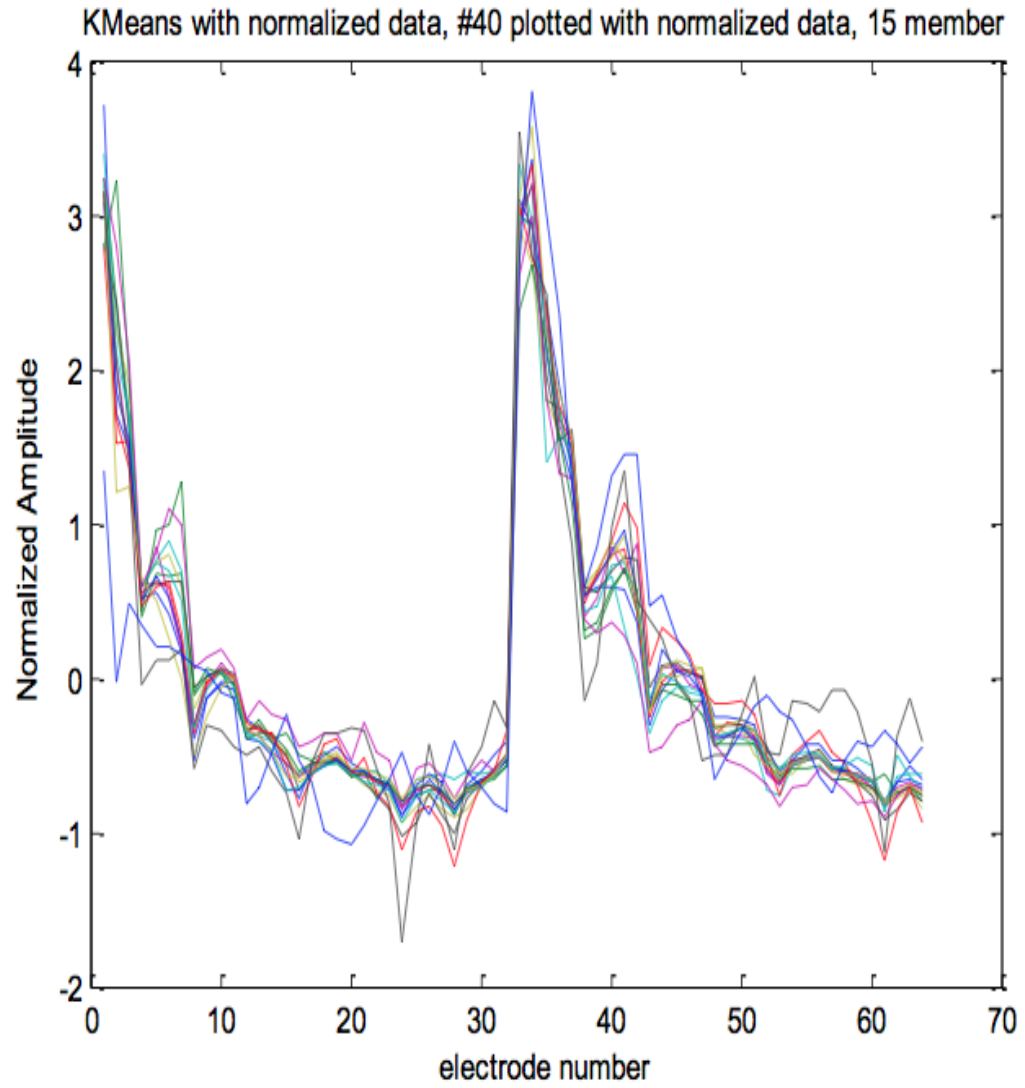


Figure 7.6: The profiles of the members in No.40 cluster in normalized data

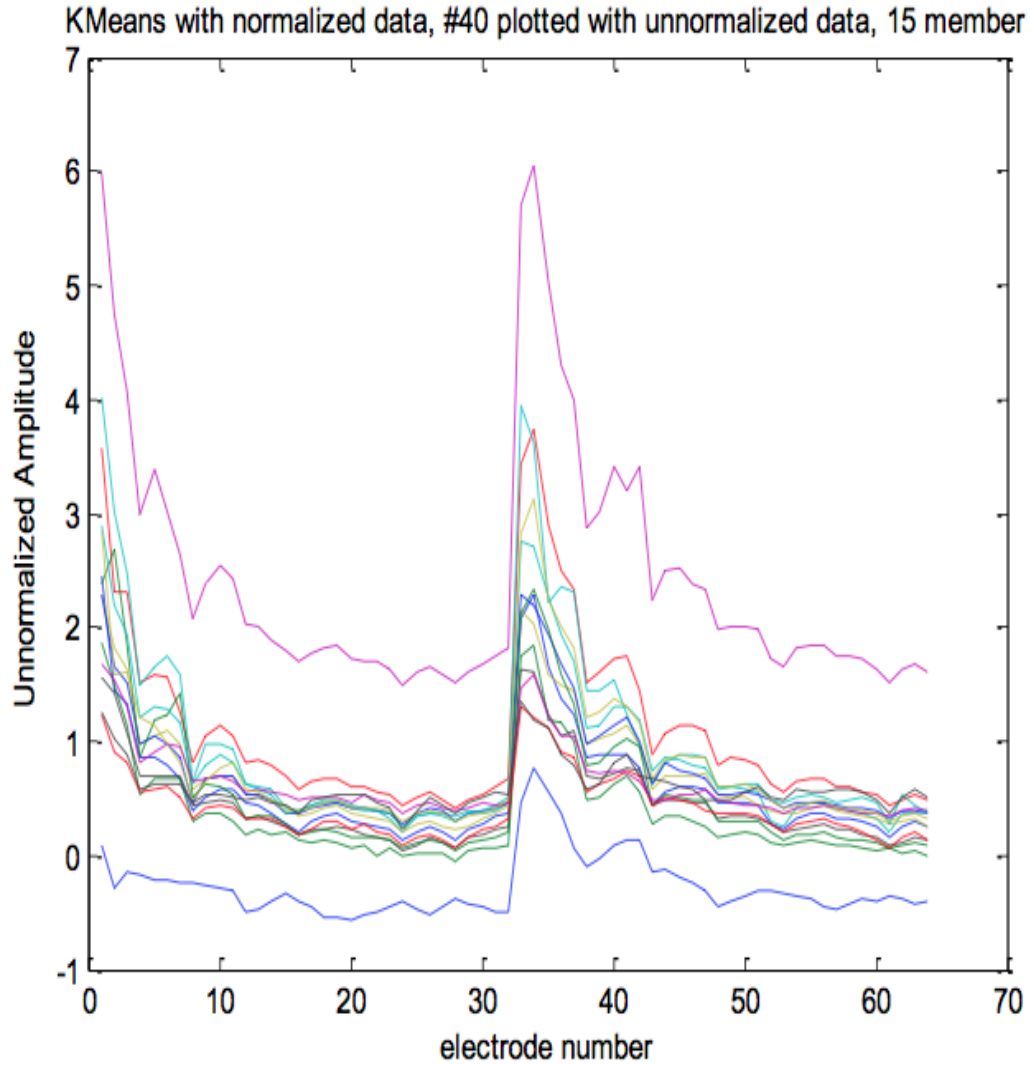


Figure 7.7: The profiles of the members in No.40 cluster in unnormalized data

We also show the largest cluster and the second largest cluster in the normalized data in Figure 7.8 and Figure 7.10, respectively. For reference, we plot the profiles of the members in these two clusters from unnormalized data in Figure 7.9 and Figure 7.11, respectively. These figures support that the normalization helps to reveal some reasonable clustering.

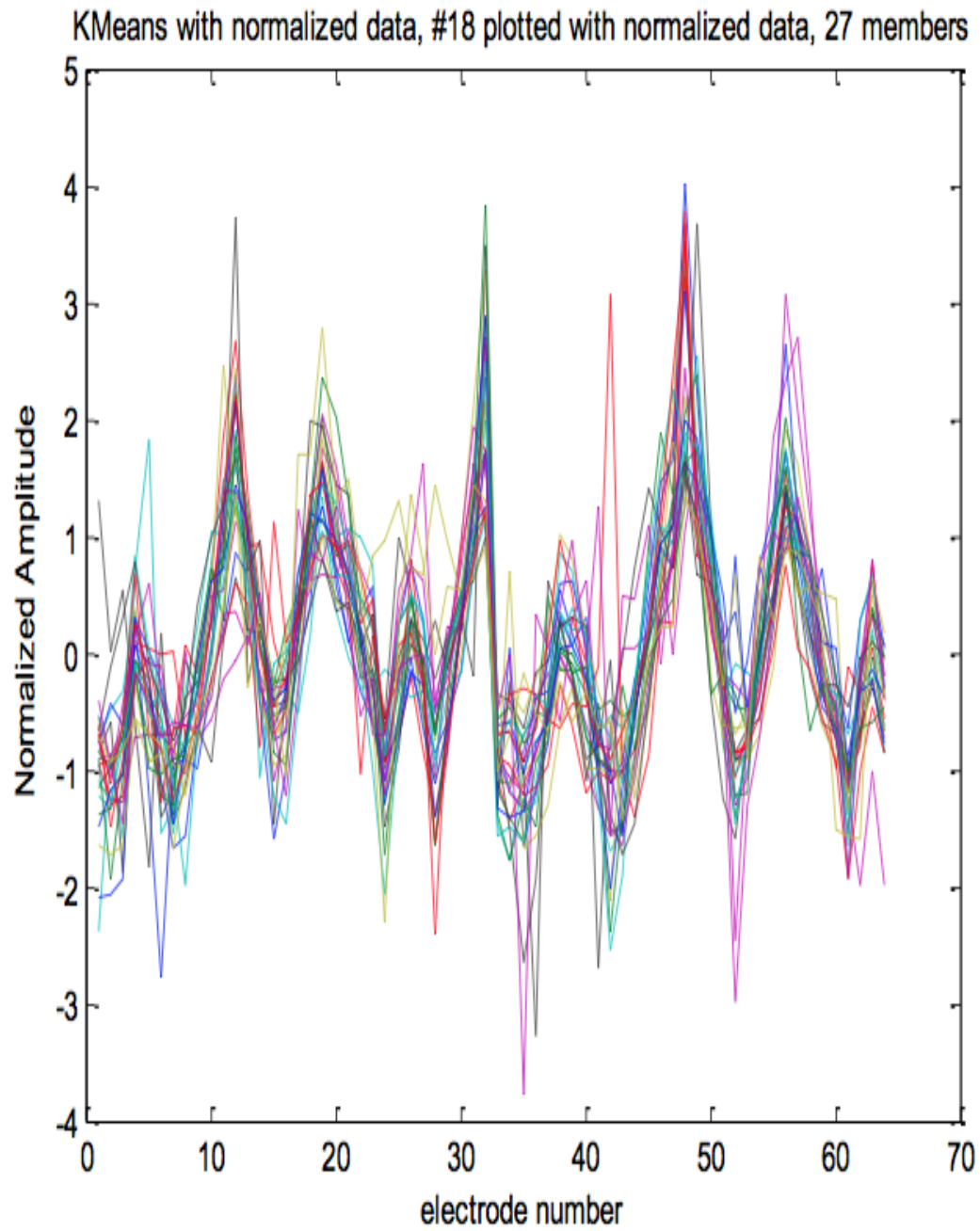


Figure 7.8: The profiles of the members in No.18 cluster in normalized data

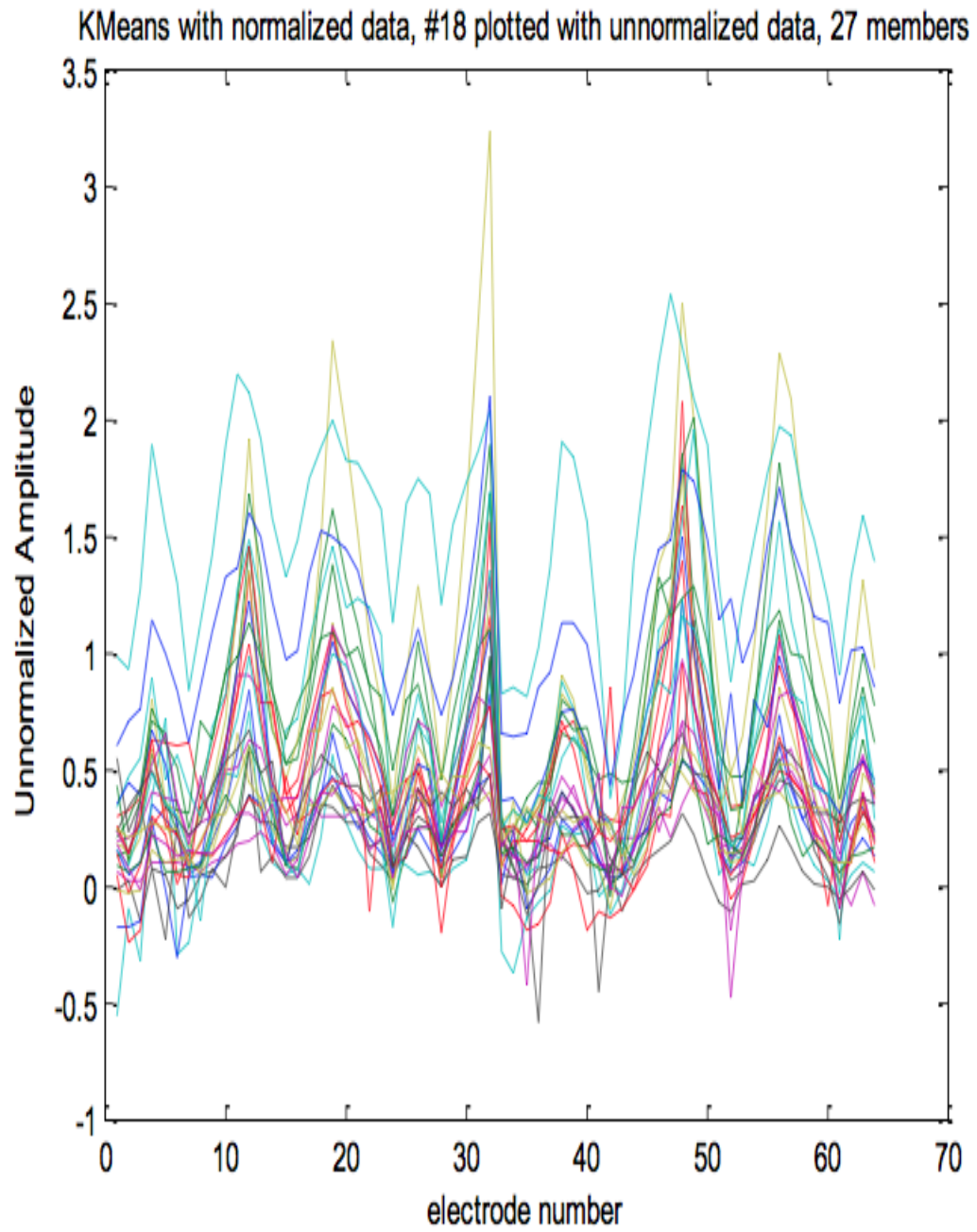


Figure 7.9: The profiles of the members in No.18 cluster in unnormalized data

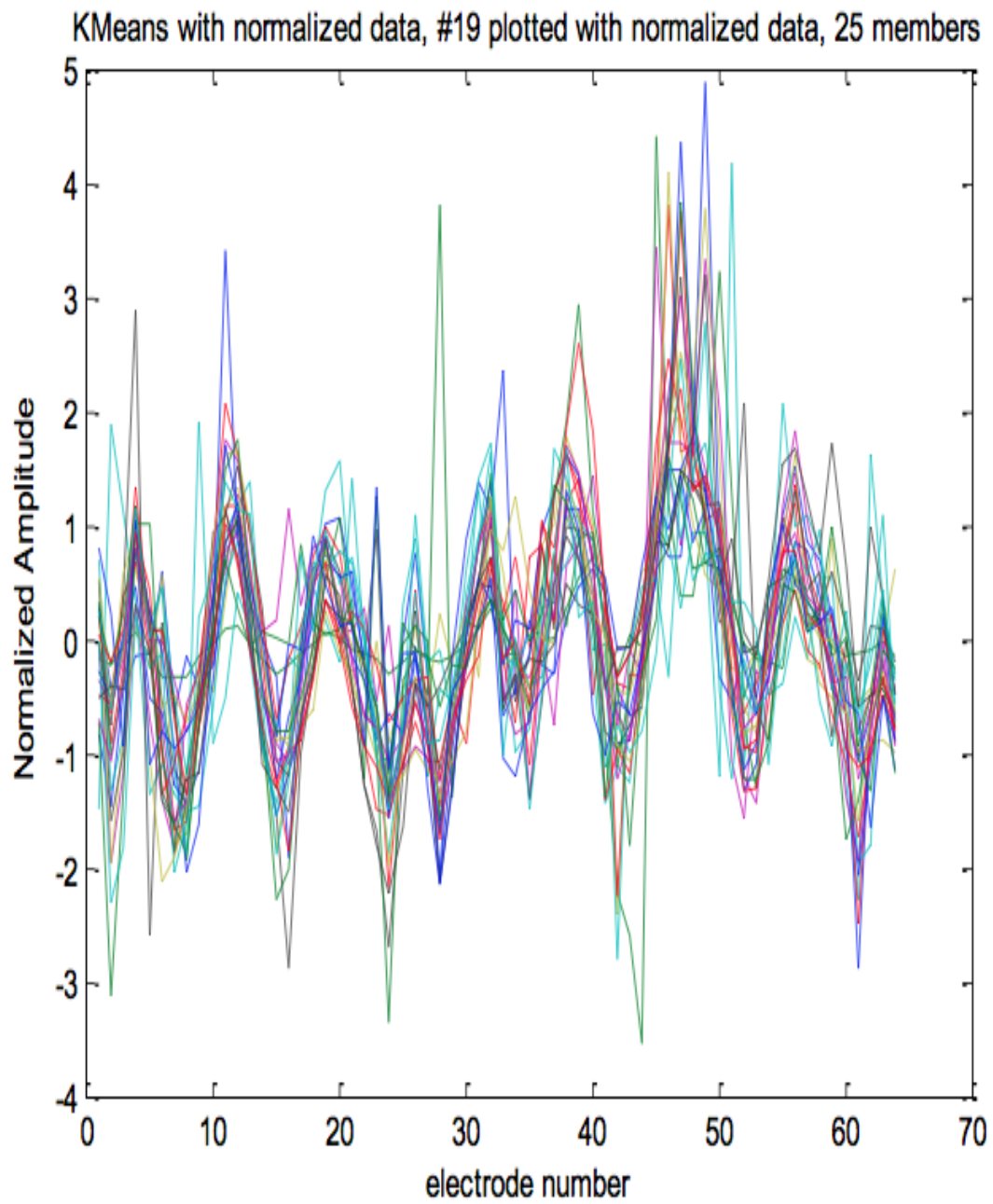


Figure 7.10: The profiles of the members in No.19 cluster in normalized data

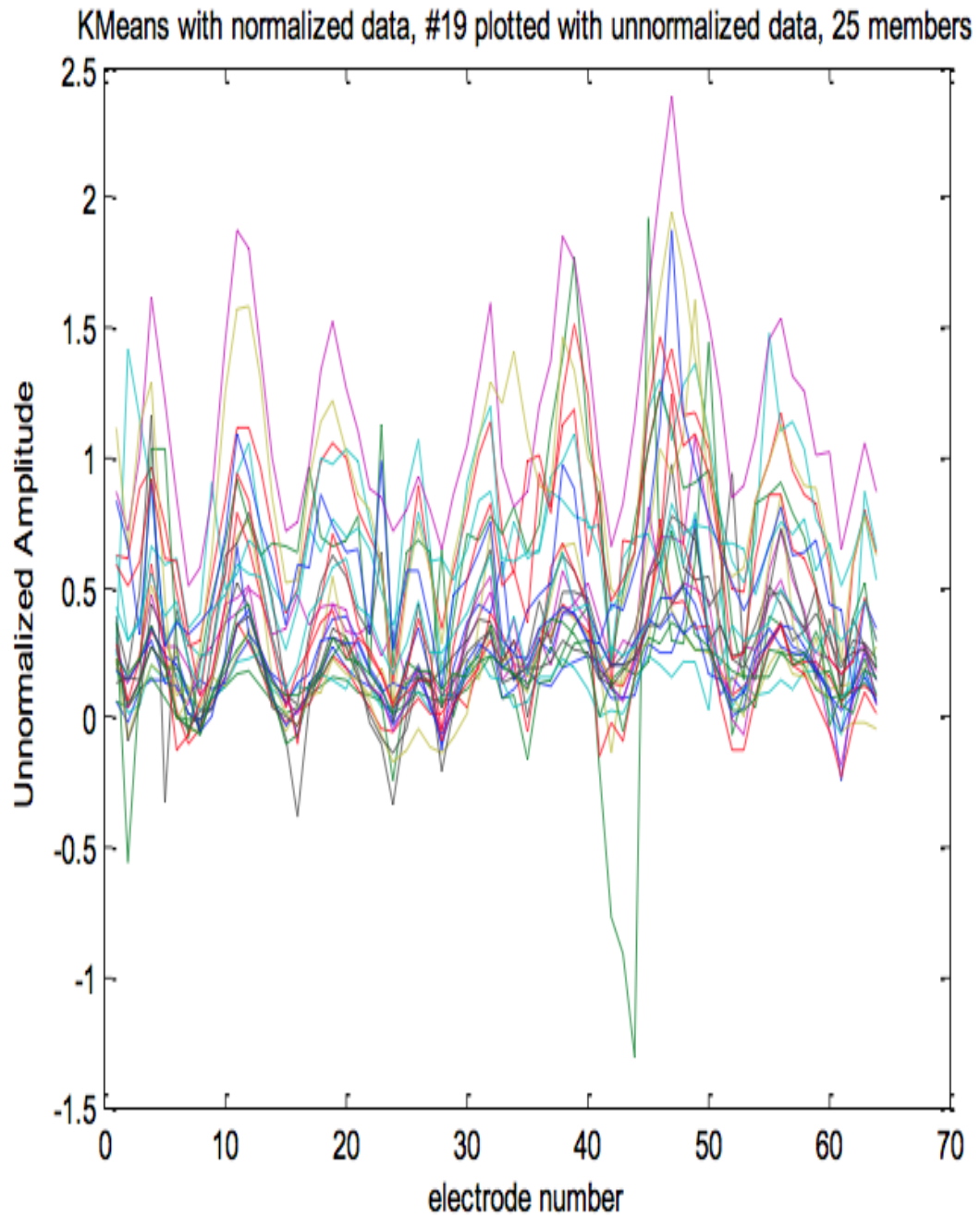


Figure 7.11: The profiles of the members in No.19 cluster in unnormalized data

Besides k-means, we apply k-medoids, HC with ward linkage, HC with complete linkage and HC with average linkage to look for  $K = 64$  clusters and  $K = 24$  clusters, as depicted in figure 7.12.



Figure 7.12: The list of clustering results we have got

Clustering Algorithm	$K=64$	$K=24$
<i>k</i> -means	√	√
<i>k</i> -medoids	√	√
HC Ward linkage	√	√
HC Complete linkage	√	√
HC Average linkage	√	√

The profiles of the members in each of these clustering for each value of  $K$  and each algorithm will be supplied separately. Furthermore, we evaluate the statistics of the number of members in the clusters for different algorithms. In figure 7.13, we show the means and standard deviations (STDs) of the number of members in the clusters for  $K = 64$  and  $K = 24$ , respectively. Apparently, the means for  $K = 64$  are same for different algorithms, which is 14 ( $896/64$ ); while the STDs are variable. Small STD represents small differences among numbers of clusters. It is the same case for  $K = 24$ .

Figure 7.13: The mean and standard deviation of the numbers of memberships

Clustering Algorithm	$K=64$	$K=24$
<i>k</i> -means	Mean 14, STD 6	Mean 37, STD 15
<i>k</i> -medoids	Mean 14, STD 8	Mean 37, STD 18
HC Ward linkage	Mean 14, STD 8	Mean 37, STD 21
HC Complete linkage	Mean 14, STD 13	Mean 37, STD 25
HC Average linkage	Mean 14, STD 27	Mean 37, STD 67

We also investigate in details how many clusters contain members (ICs) belonging to more than half subjects. As shown in figure 7.14, the results suggest nearly two thirds of clusters in both *k*-means and *k*-medoids contain members (ICs) belonging to more than seven subjects when  $K = 64$ . All of the clusters in both *k*-means and HC Ward linkage contain members (ICs) belonging to more than seven subjects when  $K = 24$ .

Figure 7.14: Number of clusters which contains ICs from more than seven ( $> 7$ ) subjects

Clustering Algorithm	$K=64$	$K=24$
<i>k</i> -means	43	24
<i>k</i> -medoids	44	23
HC Ward linkage	36	24
HC Complete linkage	28	22
HC Average linkage	17	11

### Results of ensemble Clustering

We applied both VTB and DTB binarisation techniques over the clustering results at  $K = 64$  and  $K = 24$  while varying the binarisation parameter from zero to one with 0.1 steps. For  $K = 64$ , the results of the clustering methods HC-ward, HC-average, HC-complete, *k*-means and *k*-medoids were used. For  $K = 24$ , the same five clustering methods were used in addition to the self-organising maps (SOMs) [51] and self-organising oscillator networks (SOON) [107, 6] methods. From the binarisation results produced, we have chosen a representative subset of them which shows the most reasonable results.

For  $K = 64$ , we show the results for VTB at the parameter values of 1.0, 0.8 and 0.6. Note that at the absolute tightest case (VTB 1.0), only 24 non-empty clusters out of 64 survive while including only 132 elements out of the full set of 896 elements. Although some of these clusters only have one or two elements, most of them have significant number of elements that are extremely tight and pure. It can be seen that most of these pure clusters show clear peaks at different electrodes (out of the 64 electrodes available). When widened to VTB 0.8, 36 non-empty clusters were generated including 228 elements and when widened further to VTB 0.6, 58 non-empty clusters were generated including 481 elements. It is clear that the level of tightness decreases while decreasing the VTB parameter such that some clusters at VTB 0.6 show slightly untight profiles. Though, most of the clusters at VTB 0.6 are still tight enough.

These interesting results help in focusing the research on the most useful clusters with the most useful elements instead of studying the entire set of 64 clusters with 896 elements which might include many noisy parts. Depending on the needs of the researcher, he can choose to focus at any degree of the provided tightness degrees.

For  $K = 24$ , because more clustering methods were used, the tightest cases showed completely empty clusters. The tightest useful not empty cases are provided here, which are at VTB 0.6, DTB 0.2 and DTB 0.3. One can notice that the results at  $K = 24$  are less interesting because the clusters are in general less pure than in the  $K = 64$  case. For example, at DTB 0.3, 18 non-empty clusters are generated out of 24 ones including only 102 elements. Many of the generated clusters have only one or two

elements and many of the ones that have large number of elements are not very pure. If the results of  $K = 64$  were considered for comparison for example, one can find more than 24 useful, tight and distinct clusters that collectively include only a subset of the 896 elements. This indicates that although the number of useful clusters is less than 64, a larger number of clusters must be used in order to be able to exclude the noisier elements when binarisation is tightened. Above results will be supplied separately.

The case of VTB 0.6 for  $K = 64$  and the case of DTB 0.2 for  $K = 24$  were considered for further analysis as they are the least strict cases out of the three discussed cases for each of the two values of  $K$  while still being strict enough with tight clusters. A filtering process was applied over these results' clusters such that only the clusters whose individual members represent components from more than seven subjects (at least eight subjects) are kept. The profiles for the individuals in these clusters are plotted in Figure 7.15 and Figure 7.16, for  $K = 64$  and  $K = 24$  respectively. It can be seen that under these conditions, 22 clusters including 287 elements are kept for  $K = 64$  and 15 clusters including 268 elements are kept for  $K = 24$ . Further analysis can be focused within these subsets of results.

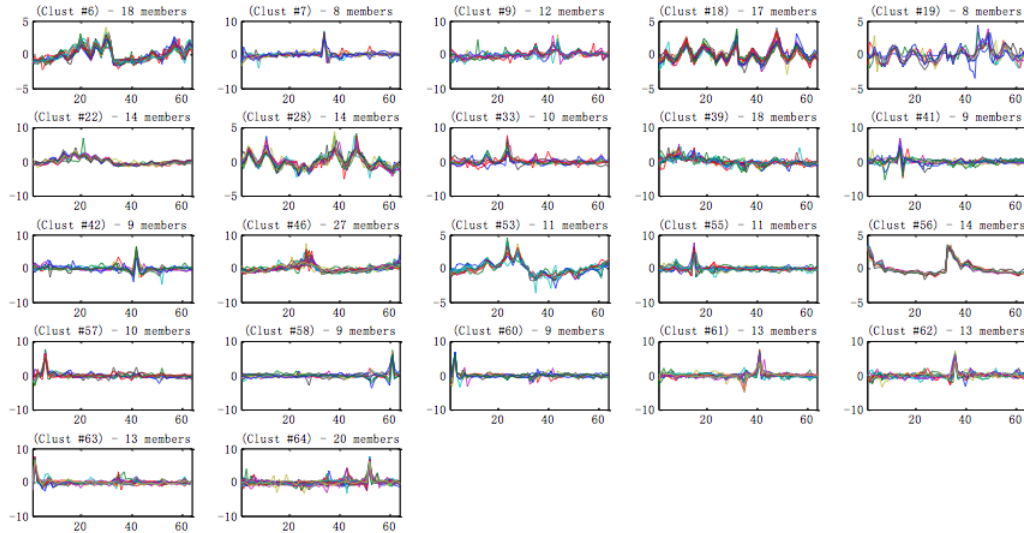


Figure 7.15: The clusters containing more than seven subjects in the ensemble clustering results:  $K = 64$

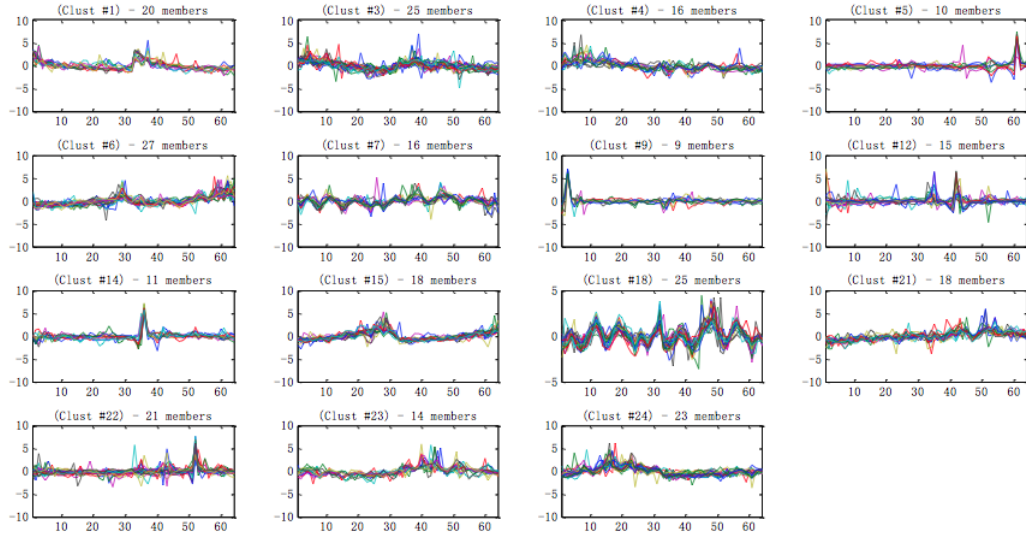


Figure 7.16: The clusters containing more than seven subjects in the ensemble clustering results:  $K = 24$

## 7.4 Conclusions

In this chapter, we outlined three clustering algorithms, namely k-means, k-medoids, and hierarchical clustering. We employed these algorithms to provide clustering on pre-processed EEG data from 14 subjects. We also explored ensemble clustering to obtain some tight clusters. Based on the results we have got, we can make some statements: firstly, normalization is necessary for this study to obtain reasonable clustering; secondly, k-means, k-medoids and HC-Ward provide relatively better clustering results; thirdly, ensemble clustering enables us to tune the tightness of the clusters so that the research can be focused.

# Chapter 8

## Summary

### 8.1 Summary and conclusions

This thesis has been devoted to the core problem of clustering and compression. Non-negative matrix factorisation (NMF) as the tool for both clustering and compression links the chapters together. Three stages of NMF have been studied throughout: NMF initialization, NMF updating strategy, NMF clustering/compression results.

Before NMF updating stage, NMF initialization is used to improve the convergence speed of the updating and the accuracy of NMF algorithm. In some cases, newly generated initial factors of NMF based on the different types of initialisation methods could be more informative than the random initial factors, and may also speed up the NMF updating. Section 3.3 has reviewed some popular NMF initialization methods based on the randomisation, clustering algorithms and the dimensionality reduction methods respectively. The random initialization among them is the most inexpensive and sometimes provides a good first estimation for NMF algorithm. Researchers also conduct the initialisation by linking to the clustering algorithms. Five clustering families rather than five clustering algorithms have been reviewed in section 2.2. Among them, k-means and Fuzzy c-means as the two common examples for cluster-based initialization have been introduced in section 3.3.2. Dimensionality reduction-based initializations based on PCA and ICA have also been reviewed in section 3.3.3. Besides these exiting initializations, chapter 4 has proposed another two NMF initializations by linking NMF with the modified k-means and IPCA respectively. To some extent, these novelty initialization methods can be employed to achieve the faster convergence and better accuracy.

For the NMF updating stage, several standard updating rules of NMF have been reviewed in section 3.2 and two novelty evolutionary optimization strategies have been proposed in chapter 5 and 6. These two proposed methods improved the iterative updating procedure of NMF by three evolving rules, aiming at producing the better cluster/compression structure. These three evolving rules have been designed, of which the first rule inherits the classical multiplicative update, while the other two rules are

driven by preserving stronger solutions offering higher quality of clusters/compression and meanwhile incorporating their altered versions to avoid local optimum, inspired by the evolutionary optimization algorithm of firefly. The proposed framework is a very general one, which can also be applied to improve NMF applications to other data analysis task by setting appropriate score function.

Effectiveness of the proposed methods in the thesis are demonstrated via careful experimental design and thorough comparative analysis using several benchmark datasets. Chapter 5 conducted the experimental evaluation with nine benchmark classification datasets from UCI machine learning repository. The NMF with the proposed evolutionary optimization strategy has been compared with the NMF with the classical multiplicative update under the different types of NMF initialisation methods. The comparison has been evaluated by the measure of the clustering performance since the goal of this chapter is to improve NMF so that it can serve better the data clustering task. Some standard cluster measures including internal evaluation and external evaluation have been previously reviewed in section 2.3 and RAND index was selected among them as an example for the cluster measure in this chapter. Experimental results have demonstrated the superior performance of the proposed method over the existing ones for data clustering evaluated with these nine benchmark datasets. Chapter 6 compressed two face image datasets saying Yale and ORL by the NMF with the proposed evolutionary optimization strategy. The proposed method has been compared with the standard one using the different measures of compression rate saying data sparsity, data orthogonality and reconstruction error. The results have shown that the proposed NMF evolutionary optimization strategy significantly improves the data compression as evaluated with these two image datasets.

Chapter 7 explored the clustering analysis of EEG dataset which is the brain response dataset. The clustering algorithms used here including k-means, k-medoids, hierarchical clustering (HC) and ensemble clustering have been previously reviewed in section 2.2. And the results show that k-means, k-medoids and HC with ward linkage provide relatively better clustering results and ensemble clustering enables us to tune the tightness of the clusters so that the research can be focused.

## 8.2 Future Works

The following is a list of possible points which could lead the continuation of the present investigation:

- Nearly all the NMF initialisations that have been developed in this thesis are based on the randomisation, clustering algorithms and the dimensionality reduction methods. Research on different types of the initialisation methods would be interesting, as these initialisations can be also embedded into our proposed NMF evolutionary optimization strategy described in chapter 5 and 6 to improve the

NMF performance.

- The evaluation of the clustering performance used in the thesis is RAND index which computes the percentage of the correct data partition offered by the clustering result as compared to the ground truth partition. However, it is not the comprehensive analysis for the cluster results and more clustering evaluations should be explored.
- In chapter 6, compression performance of the NMF with the multiplicative update has been improved by using our proposed evolutionary optimization strategy with original movement, beta movement and best movement. However, in reality, it is difficult to pre-determine the parameter  $\beta$  in beta movement for the different datasets as we do not know which  $\beta$  value can provide the better compression performance. The optimization of the parameter  $\beta$  is an important issue to be explored.
- The proposed evolutionary optimization strategies developed in chapter 5 and 6 are limited to the NMF multiplicative rule. Another NMF approximation approaches should be explored, such as alternating least square algorithms and gradient descent algorithms.
- The proposed method in chapter 6 has been developed to help the image compression task. This method could be extended to solve the image recognition task by some further studies.
- The computational cost should be considered in all the proposed methods. It would be beneficial to exploit methods with low computational complexity to obtain the significant results.

# Bibliography

- [1] Hyvarinen A. Fast and robust fixed-point algorithms for independent component analysis. *Neural Networks, IEEE Transactions on* 10, pages 626–634, 1999.
- [2] Hyvarinen A and Oja E. A fast fixed-point algorithm for independent component analysis. *Neural computation* 9, pages 1483–1492, 1997.
- [3] Raftery A. A note on bayes factors for log-linear contingency table models with vague prior information. *Journal of the Royal Statistical Society*, 48(2):249–250, 1986.
- [4] Basel Abu-Jamous, Rui Fa, David J. Roberts, and Asoke K. Nandi. Paradigm of tunable clustering using binarization of consensus partition matrices (bi-copam) for gene discovery. *PLOS ONE Journal*, 8(2):1–14, 2013.
- [5] R. Avogadri and G. Valentini. Ensemble clustering with a fuzzy approach. *Studies in Comput. Intell. Superv. and Unsuperv. Ensemble Methods their Appl.*, 126:49–69, 2008.
- [6] H. G. Ayad and M. S. Kamel. On voting-based consensus of cluster ensembles. *Pattern Recognition*, 43:1943–C1953, 2010.
- [7] M. W. Berrya, M. B. Amy, N. Langvilleb, V. Paul Paucac, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52:155–173, 2007.
- [8] P. Bertone and M. Gerstein. Integrative data mining: the new direction in bioinformatics. *Engineering in Medicine and Biology Magazine, IEEE*, 20(4):33–40, 2001.
- [9] Bezdek. Pattern recognition with fuzzy objective function algorithms. *New York: Plenum*, 1981.
- [10] James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Comput. Geosoci*, 10(2-3):191–203, 1984.



- [11] G. Bin, W. L. Woo, and B. W. K. Ling. Improving pomdp tractability via belief compression and clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(1):125–136, 2010.
- [12] C. Boutsidisa and E. Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41:1350–1362, 2008.
- [13] J P Brunet, P Tamayo, T R Golub, and J P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 101(12), pp. 4164- 4173, 2004.
- [14] Ding C, He XF, and Simon HD. On the equivalence of nonnegative matrix factorization and spectral clustering. *In: Kargupta H, Srivastava J, Kamath C, Goodman A, editors. Proceedings of the SIAM international conference data mining (SDM05). Philadelphia: Society for Industrial and Applied Mathematics*, pages 606–616, 2005.
- [15] Jutten C and Herault J. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Process* 24, pages 1–10, 1991.
- [16] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.
- [17] Lin Chih-Jen. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6), 2007.
- [18] Lin Chih-Jen. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.
- [19] A. Cichocki, R. Zdunek, A. H. Phan, and S. I. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley, 2009.
- [20] FY Cong, V Alluri, AK Nandi, P Toivainen, Rui Fa, Basel Abu-Jamous, Liyun Gong, BGW Craenen, H Poikonen, M Huutilainen, and T Ristaniemi. Linking brain responses to naturalistic music through analysis of ongoing eeg and stimulus features. *IEEE Transactions on Multimedia*, 15(5):1060–1069, 2013.
- [21] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224227, 1979.
- [22] P. Day and A. K. Nandi. Robust text-independent speaker verification using genetic programming. *IEEE Transaction on Audio, Speech and Language Processing*, 15:285–295, 2007.

- [23] Lee DD and Seung HS. Learning the parts of objects by non- negative matrix factorization. *Nature*, 401:788–791, 1999.
- [24] H.S. Seung D.D. Lee. Algorithms for non-negative matrix factorization. In *Adv. Neural Inf. Process. Systems 13 (2000)*, pages 556–562, 2000.
- [25] M. A. de L. Balaguer and C. M. Williams. A cluster validity framework based on induced partition dissimilarity. *IEEE Transactions on Cybernetics*, 43:308–320, 2013.
- [26] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. in *Proc. Tenth ACM SIGKDD Int. Conf. Know. disc. data mining, New York, NY, USA, KDD 04,ACM*, pages 551–556, 2004.
- [27] E. Diday and J. C. Simon. Cluster analysis. in *Digital Pattern Recognition, Berlin, Springer-Verlag*, 1976.
- [28] R. Dubes and A. K. Jain. Cluster methodologies in exploratory data analysis. *Advances in Comput.*, 19:113–228, 1980.
- [29] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *J. Cyber.*, 3(3):32–57, 1973.
- [30] Darius M. Dziuda. *Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data*. Wiley, 2010.
- [31] B. Everitt. Cluster analysis. *Heinemann Education*, 1974.
- [32] Bach F and Jordan M. Kernel independent component analysis. *journal of Machine Learning Research*, 3:1–48, 2002.
- [33] R. Fa and A. K. Nandi. Parametric validity index of clustering for microarray gene expression data. in *IEEE Int. Workshop Machine Learning for Sig. Process*, 2011.
- [34] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. in *Proc. Int. Conf. Tools with Art. Intell.*, pages 418–426, 2003.
- [35] Imola K. Fodor. A survey of dimension reduction techniques. *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory P.O. Box 808, L-560, Livermore, CA 94551*, 2002.
- [36] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.
- [37] A. Fred and A. K. Jain. Data clustering using evidence accumulation. in *Proc. Sixteenth Int. Conf. Pattern Recognition (ICPR)*, pages 276–280, 2002.

- [38] Bin Gao, W.L. Woo, and B.W.-K. Ling. Machine learning source separation using maximum a posteriori nonnegative matrix factorization. *IEEE Transactions on Cybernetics*, 44:1169–1179, 2014.
- [39] Y Gao and G Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, vol. 21, pp. 3970-3975, 2005.
- [40] B. Gokberk, H. Dutagaci, A. Ulas, and L. Akarun. Representation plurality and fusion for 3-d face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(1):155–173, 2008.
- [41] D Guillamet, J Vitria, and B Scheile. Introducing a weighted nonnegative matrix factorization for image classification. *Pattern Recognition Letters*, 24:2447–2454, 2003.
- [42] H. Hae-Jin, P. Yi, R. Harrison, and P. C. Tai. Improved protein secondary structure prediction using support vector machine with a new encoding scheme and an advanced tertiary classifier. *IEEE Trans. on NanoBioscience*, 3(4):265–271, 2004.
- [43] Oja E Hyvarinen A. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.
- [44] A. K. Jain and R. C. Dubes. Algorithms for clustering data. *Englewood Cliffs, NJ: Prentice-Hall*, 1988.
- [45] M. Jambu. Fortran iv computer program for rapid hierarchical classification of large data sets. *Computers and Geosciences*, 7(3):297–310, 1981.
- [46] Pujar J.H. and Kadlaskar L.M. A new lossless method of image compression and decompression using huffman coding techniques. *Journal of Theoretical and Applied Information Technology*, 15(1):18–23, 2010.
- [47] Y. Jian, A. F. Frangi, J.-Y. Yang, D. Zhang, and Z. Jin. Kpca plus lda: a complete kernel fisher discriminant framework for feature extraction and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(2):230–244, 2005.
- [48] D. X. Jiang, C. Tang, and A. D. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. Know. and Data Eng.*, 16(11):1370–1386, 2004.
- [49] Brunet JP, Tamayo P, Golub TR, and Mesirov JP. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12):4164–4173, 2004.
- [50] Du K-L and Swamy MNS. Independent component analysis. neural networks and statistical learning. *Springer London.*, pages 419–450, 2014.

- [51] T. E. Kohonen. Self-organizing maps. *New York: Springer-Verlag*, 1997.
- [52] Kaufman L. and Rousseeuw P.J. Clustering by means of medoids, in statistical data analysis based on the  $l_1$ -norm and related methods. *edited by Y. Dodge, North-Holland*, pages 405–416, 1987.
- [53] Kaufman L. and Rousseeuw P.J. Finding groups in data: An introduction to cluster analysis. *Wiley, Canada*, 1990.
- [54] Benson S. Y. Lam and Hong Yan. Assessment of microarray data clustering results based on a new geometrical index for cluster validity. *Soft Computing*, 11(4):341–348, 2007.
- [55] A N Langville, C D Meyer, and R Albright. Initializations for the nonnegative matrix factorization. *Proceeding of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.
- [56] S.Z. Li, X.W. Hou, H.J. Zhang, and Q.S Cheng. Learning spatially localized parts-based representation. *In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 207–212, 2001.
- [57] C. Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596, 2007.
- [58] C. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.
- [59] Cheng-Ru Lin and Ming-Syan Chen. Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Trans. Know. Data Eng.*, 17(2):145–159, 2005.
- [60] C.-L. Liu, S. Jaeger, and M. Nakagawa. Online recognition of chinese characters: the state-of-the-art. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):198–213, 2004.
- [61] W Liu, K Yuan, and D Ye. Reducing microarray data via nonnegative matrix factorization for visualization and clustering analysis. *Journal of Biomedical Informatics*, 41:602–606, 2008.
- [62] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu. Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, 43:982–994, 2013.
- [63] Z. G. Liu, D. C. Chen, and H. Bensmail. Gene expression data classification with kernel principal component analysis. *J. Biomed. Biotech.*, 2005(2):155–159, 2005.

- [64] S. P. Lloyd. Least squares quantization in pcm, technical report, bell laboratories. *IEEE Transactions on Information Theory* 28, pages 129–137, 1957.
- [65] Scholz M, Gatzek S, Sterling A, Fiehn O, and Selbig J.: Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics* 2004, 20(15):2447–2454, 2004.
- [66] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Berkeley, CA: University of California Press*, pages 281–297, 1967.
- [67] Y. Mao, X. B. Zhou, D. Y. Pi, Y. X. Sun, and Stephen T. C. Wong. Multiclass cancer classification by using fuzzy support vector machine and binary decision tree with gene selection. *J. Biomed. Biotech.*, (2):160–171, 2005.
- [68] Llus Marquez. Machine learning and natural language processing. *Technical Report LSI-00-45-R, Department de Llenguatges i Sistemes Informatics (LSI), University Politecnica de Catalunya (UPC)*, 2000.
- [69] D. L. Massart, F. Plastra, and L. Kaufman. Non-hierarchical clustering with masloc. *Pattern Recognition*, 16:507–516, 1983.
- [70] U Maulik and S Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1650–1654, 2002.
- [71] T. M. Mitchell. Machine learning. *McGraw Hill*, 1997.
- [72] T. M. Mitchell. The discipline and future of machine learning. machine learning. *Department, School of Computer Science, Carnegie Mellon University*, 2007.
- [73] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2003.
- [74] R. Navigli, P. Velardi, and A. Gangemi. Ontology learning and its application to automated terminology translation. *Intelligent Systems, IEEE*, 18(1):1541–1672, 2003.
- [75] S. Nikitidis, A. Tefas, and I. Pitas. Projected gradients for subclass discriminant nonnegative subspace learning. *IEEE Transactions on Cybernetics*, 2014. In press, DOI: 10.1109/TCYB.2014.2317174.
- [76] Nils J. Nilsson. Introduction to machine learning. *Artificial Intelligence Laboratory, Department of Computer Science, Stanford University*, 2005.

- [77] Mahdi O.A., Mohammed M.A., and Mohamed A.J. Implementing a novel approach an convert audio compression to text coding via hybrid technique. *International Journal of Computer Science*, 9(6):53–59, 2013.
- [78] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.
- [79] A. D. Parkins and A. K. Nandi. Genetic programming techniques for hand written digit recognition. *Signal Processing*, 84(12):2345–2365, 2004.
- [80] A. D. Parkins and A. K. Nandi. Method for calculating first-order derivative based feature saliency information in a trained neural network and its application to handwritten digit recognition. *IEE Proceedings - Part VIS*, 152(2):137–147, 2005.
- [81] R.D. Pascual-Marqui, A.D. Pascual-Montano, K. Kochi, and J.M. Carazo. Smoothly distributed fuzzy c-means: a new self-organizing map. *Pattern Recognition*, 34(12):2395–2402, 2001.
- [82] A Pascual-Montano, P Carmona-Saez, M Chagoyen, F Tirado, J M Carazo, and R D Pascual-Marqui. bionmf: a versatile tool for nonnegative matrix factorization in biology. *BMC Bioinformatics*, vol. 7(1), pp. 366-374, 2006.
- [83] J. M. Pena, J. A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for k-means algorithm. *Elsevier Science B. V, Pattern Recognition Letters 20*, 20(10):1027–1040, 1999.
- [84] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.
- [85] Press, W.H., S.A.Teukolsky, W.T. Vetterling, and B.P.Flannery. Numerical recipes: the art of scientific computing. *New York: Cambridge University Press*, 2007.
- [86] Jie Qin, Darrin P.Lewis, and William Stafford Noble. Kernel hierarchical gene clustering from microarray expression data. *Bioinformat.*, 19(16):2097–2104, 2003.
- [87] W M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association, Theory and Methods Section*, 66:846–850, 1971.
- [88] M Rezaei and R Boostani. An efficient initialization method for nonnegative matrix factorization. *Journal of Applied Sciences*, 11:354–359, 2011.

- [89] Dhillon I. S. and Modha D. M. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.
- [90] Cruces-Alvarez SA and Amari S Cichocki A. From blind signal extraction to blind instantaneous signal separation: criteria, algorithms, and stability. *IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council* 15, pages 859–873, 2004.
- [91] Sameh A. Salem, Lindsay B. Jack, and Asoke K. Nandi. Investigation of self-organizing oscillator networks for use in clustering microarray data. *IEEE Trans. NanoBio.*, 7(1):65–79, 2008.
- [92] Sameh A. Salem and Asoke K. Nandi. Development of assessment criteria for clustering algorithms. *Pattern Anal. and Appl.*, 12(1):79–98, 2009.
- [93] R. Schachtner. *Extensions of non-negative matrix factorization and their application to the analysis of wafer test data*. PhD thesis, University of Regensburg, 2010.
- [94] F. Shahnaza, M. W. Berrya, V. P. Paucab, and R. J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing and Management*, 42(2):373–386, 2006.
- [95] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way Analysis*. Wiley, West Sussex, England, 2004.
- [96] M. Srinivas and L.M. Patnaik. Genetic algorithms: a survey. *IEEE Computer*, 27(6):17–26, 1994.
- [97] A. Strehl and J. Ghosh. Cluster ensembles c a knowledge reuse framework for combining multiple partitions. *J. Machine Learning Research*, 3:583–617, 2002.
- [98] Stephen Swift, Allan Tucker, Veronica Vinciotti, Nigel Martin, Chris tine Orengo, Xiaohui Liu, and Paul Kellam. Consensus clustering and functional interpretation of gene-expression data. *Genome Biology*, 5(11):R94, 2004.
- [99] Hastie T and Tibshirani R. Independent components analysis through product density estimation. *Neural Information Processing Systems (NIPS)*, 2002.
- [100] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitro-vsky, E. S. Lander, and T. R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Nat. Academy Sci. USA*, 96(6):2907–2912, 1999.

- [101] Shuta Tomida, Taizo Hanai, Hiroyuki Honda, and Takeshi Kobayashi. Analysis of expression profile using fuzzy adaptive resonance theory. *Bioinformat.*, 18(8):1073–1083, 2002.
- [102] H. Tong-Cheng and D. You-Dong. Generic object recognition via integrating distinct features with svm. *In Proc. of the Intl Conf. on Machine Learning and Cybernetics*, pages 3897–3902, 2006.
- [103] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper. Weighted cluster ensemble using a kernel consensus function. *Lecture Notes in Computer Science, Heidelberg: Springer.*, 5197:195–202, 2008.
- [104] S. Vega-Pons and J. Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25:337–372, 2011.
- [105] G Wang, A V Kossenkov, and M F Ochs. Ls-nmf: a modified nonnegative matrix factorization algorithm utilizing uncertainty estimates. *BMC Bioinformatics*, 7:175, 2006.
- [106] W. Wang. An improved non-negative matrix factorization algorithm for combining multiple clusterings. *in 2010 Int. Conf. Machine Vision Human-machine Interface*, pages 604–607, 2010.
- [107] A. Weingessel, E. Dimitriadou, and K. Hornik. An ensemble method for clustering. *in Distributed Statistical Computing, Vienna*, 2003.
- [108] Darrell Whitley and Andrew M. Sutton. Genetic algorithms - a survey of models and methods. *Springer Berlin Heidelberg*, 21:637–671, 2012.
- [109] S Wild. Seeding non-negative matrix factorizations with the spherical k-means clustering. *Master of Science Thesis, University of Colorado*, 2003.
- [110] S. Winters-Hilt, M. Landry, M. Akesson, M. Tanase, I. Amin, A. Coombs, E. Morales, J. Millet, C. Baribault, and S. Sendamangalam. Cheminformatics methods for novel nanopore analysis of hiv dna termini. *BMC Bioinformatics*, 7 Suppl 2: S22, 2006.
- [111] Shuanhu Wu, A.W.-C. Liew, Hong Yan, and Mengsu Yang. Cluster analysis of gene expression data based on self-splitting and merging competitive learning. *IEEE Trans. Inf. Tech. Biomed.*, 8(1):5–15, 2004.
- [112] Yang X.S. Firefly algorithm, stochastic test functions and design optimization. *Int. J. bio-inspired computation*, 2010.



- [113] R. Xu and D. II Wunsch. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3):645–678, 2005.
- [114] W. Xu, A. K. Nandi, and J. Zhang. Novel fuzzy reinforced learning vector quantisation algorithm and its application in image compression. *IEEE Proceedings - Part VIS*, 150(5):292–298, 2003.
- [115] Yun Xue, Chong Sze Tong, Ying Chen, and Wen-Sheng Chen. Clustering-based initialization for non-negative matrix factorization. *Applied Mathematics and Computation 205*, pages 525–536, 2008.
- [116] S. Yang, J. Song, H. Rajamani, C. Taewon, Y. Zhang, and R. Mooney. Fast and effective worm fingerprinting via machine learning. *In Proc. of the Intl Conf. on Autonomic Computing, ICAC, TX, US*, pages 311–313, 2006.
- [117] F Z Yao, J Coquery, and K A Le Cao. Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets. *BMC Bioinformatics*, 13:24:1471–2105, 2012.
- [118] Fangzhou Yao, Jeff Coquery, and Kim-Anh L Cao. Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets. *BMC Bioinformatics*, 13:24:1471–2105, 2012.
- [119] Shaohui Yu, Yujun Zhang, Wenqing Liu, Nanjing Zhao, Xue Xiao, and Gaofang Yin. A novel initialization method for nonnegative matrix factorization and its application in component recognition with three-dimensional fluorescence spectra. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 86:315–319, 2012.
- [120] Ya-Jun Zhang and Zhi-Qiang Liu. Self-splitting competitive learning: a new online clustering paradigm. *IEEE Trans. Neural Networks*, 13(2):369–380, 2002.
- [121] L Zhao, G Zhuang, and X Xu. Facial expression recognition based on pca and nmf. *Proceedings of the 7th World Congress on Intelligent Control and Automation*, pages 6826–6829, 2008.
- [122] Z Zheng, J Yang, and Y Zhu. Initialization enhancer for non-negative matrix factorization. *Engineering Applications of Artificial Intelligence*, 20:101–110, 2007.

