Extreme learning machine for classification and clustering on multiview data

Chen, Jichao

2021

Chen, J. (2021). Extreme learning machine for classification and clustering on multiview data. Doctoral thesis, Nanyang Technological University, Singapore.
<https://hdl.handle.net/10356/153094>

<https://hdl.handle.net/10356/153094>

<https://doi.org/10.32657/10356/153094>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:40:00 SGT

Extreme Learning Machine for Classification and Clustering on Multiview Data

Chen Jichao

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

2 Nov 2021
.....

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
.....

Chen Jichao

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

03 Nov. 2021

.....

Date

.....

Lin Zhiping

Authorship Attribution Statement

This thesis contains material from 4 papers published in the following peer-reviewed journals / from papers accepted at conferences in which I am listed as the first author.

Chapter 3 is published as Jichao Chen, Yijie Zeng, Yue Li, Guang-Bin Huang. "Octree-based Convolutional Autoencoder Extreme Learning Machine for 3D Shape Classification". 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018, pp. 1-7, doi: 10.1109/IJCNN.2018.8489409.

The contribution of the co-authors are as follows:

- Prof. Huang Guang-Bin provided the initial direction and supervised the work. He also reviewed the manuscript draft.
- I proposed the main algorithm and performed all the experiments at the School of Electrical and Electronic Engineering. I also analyzed the results.
- I prepared the manuscript draft.
- Dr. Zeng Yijie assisted in the implementation of the proposed algorithm.
- Dr. Zeng Yijie, Dr. Wang Siqi and Dr. Soh Ling Min reviewed the manuscript draft.

Chapter 4 is published as Jichao Chen, Yijie Zeng, Yue Li, Guang-Bin Huang. "Unsupervised Feature Selection based Extreme Learning Machine for Clustering". in: Neurocomputing (2019). doi: 10.1016/j.neucom.2019.12.065.

The contribution of the co-authors are as follows:

- Prof. Huang Guang-Bin provided the initial direction and supervised the work. He also reviewed the manuscript.
- I proposed the main algorithm and performed all the experiments at the School of Electrical and Electronic Engineering. I also analyzed the results.
- I prepared the manuscript draft. The draft was reviewed together with Dr. Zeng Yijie, Dr. Li Yue.
- Dr. Zeng Yijie assisted in the implementation of the proposed algorithm.
- Dr. Zeng Yijie and Dr. Li Yue assisted in the analysis of the proposed algorithms.

Chapter 5 is submitted to Neurocomputing as Jichao Chen, Yijie Zeng, Guang-Bin Huang. "Distance Fusion for Multiview Clustering". Under review.

The contribution of the co-authors are as follows:

- Prof. Huang Guang-Bin provided the initial direction and supervised the work. He also reviewed the manuscript.
- I proposed the main algorithm and performed all the experiments at the School of Electrical and Electronic Engineering. I also analyzed the results.
- I prepared the manuscript draft. The draft was reviewed together with Dr. Zeng Yijie.
- Dr. Zeng Yijie assisted in the implementation of the proposed algorithm.

Chapter 6 is published as Jichao Chen, Guang-Bin Huang. "Dual Distance Adaptive Multiview Clustering". in: Neurocomputing (2021). doi: 10.1016/j.neucom.2021.01.132.

The contribution of the co-authors are as follows:

- Prof. Huang Guang-Bin provided the initial direction and supervised the work. He also reviewed the manuscript.
- I proposed the main algorithm and performed all the experiments at the School of Electrical and Electronic Engineering. I also analyzed the results.

2 Nov 2021

.....
Date

.....

Chen Jichao

Acknowledgements

I wish to express my gratitude to my supervisors Professor Lin Zhiping and Professor Huang Guangbin for their guidance. I would like to thank Dr. Zeng Yijie for his advice and support. I would like to thanks Professor Cao Qi for his suggestions and support. This thesis would never be possible without you.

Abstract

Many real-world applications deal with data that have multiple feature sets. Such data is called multiview data, where each feature set is considered as a view. Many machine learning tasks have their multiview versions, which have similar objectives but dealing with multiview data. For example, the most general classification and clustering tasks have their multiview versions. Multiview classification aims to categorize multiview data into predefined classes. It is quite useful when dealing with complicated objects presented in different formats (e.g., 3D models). Multiview clustering separates the multiview data into clusters by considering their pairwise similarities in all views. It is widely used in feature learning and data labeling. Multiview spectral clustering is one of the most popular kinds of multiview clustering algorithms, which leverages eigendecomposition on the graph Laplacian to find the clusters. Meanwhile, extreme learning machine (ELM) is attracting more and more attention for its promising performance in various real-world applications. Therefore, the thesis investigates ELM for classification and clustering on multiview data.

Firstly, the thesis investigates multiview 3D shape classification. The thesis extends the idea of ELM's random neurons to octree based operations and proposes octree based convolutional autoencoder extreme learning machine (OCA-ELM) for 3D shape classification. The ELM auto-encoder (ELM-AE) is adopted for feature extraction and feature fusion, and an ELM classifier is utilized for the final classification. The experimental results prove that the input and operations based on octree can improve the classification performance.

Secondly, the thesis investigates spectral clustering with ELM. ELM is a training algorithm for single-layer feedforward networks, famous for its nonlinear feature mapping. The thesis develops a clustering algorithm called unsupervised feature selection based extreme learning machine (UFS-ELM). The proposed algorithm first transfers the data into a hidden space with nonlinear feature mapping of the ELM. It reduces the dimensionality of features to the desired number of clusters

while preserving the graph structure. An unsupervised feature selection method is embedded in the formulation to eliminate the worthless hidden neurons. With a simple constraint, the method can directly output the clustering result without post-processing. Experimental results demonstrate the effectiveness of UFS-ELM on a wide range of datasets.

Thirdly, the thesis explores spectral clustering on multiview data. The views of multiview data might have different data types, scales, and data distributions. The information of multiple views needs to be fused into a single one at some moment to derive a consistent clustering result across different views. Existing multiview spectral clustering methods fuse multiple similarity matrices or Laplacian matrices into a common matrix. This thesis proposes a term of distance fusion that fuses the information from the distance perspective. Specifically, the proposed method provides a comprehensive distance measure of a unified scale. A common distance can be produced by combining multiple distance matrices without weighting. With the distance fusion, the thesis presents a novel multiview spectral clustering method called distance fusion with adaptive neighbors (DFAN). An empirical study on several multiview datasets shows that the proposed DFAN method surpasses the state-of-the-art multiview spectral clustering methods. The experimental results also show that the proposed method enlarges the distances between different clusters and is robust to views with misleading information.

Finally, the thesis proposes to incorporate extreme learning machine with multiview spectral clustering. Existing multiview spectral clustering methods construct a similarity matrix based on the pairwise distances measured in the original data space or linearly projected space. However, the actual data structure might lie on a complex nonlinear manifold. In this thesis, the extreme learning machine is adopted to learn a nonlinearly mapped embedding space where the intrinsic structure is preserved. The distances measured in both embedding space and the original space can be fused to derive a more robust distance measure for each view. Massive experiments demonstrate the proposed multiview clustering method, termed dual distance adaptive multiview clustering (DAMC), is superior to the state-of-the-art multiview clustering methods.

Contents

Acknowledgements	ix
Abstract	xi
List of Figures	xvii
List of Tables	xix
Symbols and Acronyms	xxi
1 Introduction	1
1.1 Background	1
1.2 Contributions	3
1.3 Organization	4
2 Literature Review	7
2.1 Clustering	7
2.1.1 Spectral Clustering	8
2.1.2 Constrained Laplacian Rank	9
2.2 Unsupervised Feature Selection	10
2.3 Extreme Learning Machine	11
2.3.1 Extreme Learning Machine for Clustering	14
2.4 Multiview Clustering	15
2.4.1 Multiview Spectral Clustering	19
2.4.2 Multiview Learning with Adaptive Neighbors	20
2.5 3D Shape Classification	21
3 Octree based Convolutional Autoencoder Extreme Learning Machine for 3D Shape Classification	23
3.1 Background and Motivation	24
3.2 Methodology	25
3.2.1 Input Representations	26
3.2.2 Feature Map Extraction	27
3.2.3 Feature Extraction	27

3.2.4	Classification	29
3.3	Experiments	31
3.3.1	Datasets	31
3.3.2	3D Shape Classification	32
3.4	Summary	36
4	Unsupervised Feature Selection based Extreme Learning Machine for Clustering	37
4.1	Background and Motivation	38
4.2	Unsupervised Feature Selection based Extreme Learning Machine for Clustering	39
4.2.1	Objective Function	39
4.2.2	Alternating Optimization	41
4.3	Experimental Results and Discussions	42
4.3.1	Datasets	42
4.3.2	Performance metric	45
4.3.3	Experimental Setup	46
4.3.4	Comparison with Related Algorithms	46
4.3.5	Computational Complexity	52
4.3.6	Influence of Trade-off Parameters	53
4.3.7	Influence of Number of Hidden Neurons	55
4.3.8	Influence of L2,1 Norm	57
4.4	Summary	58
5	Distance Fusion for Multiview Clustering	59
5.1	Background and Motivation	60
5.2	Proposed Methods	61
5.2.1	Distance Fusion	61
5.2.2	Distance Fusion with Adaptive neighbors	63
5.3	Optimization	64
5.3.1	Update F with fixed A	65
5.3.2	Update A with fixed F	65
5.3.3	Initialization	66
5.3.4	Parameters and Stopping Criterion	67
5.4	Experiments	67
5.4.1	Datasets	68
5.4.1.1	Handwritten Digits	71
5.4.1.2	MSRCv1	71
5.4.1.3	COIL20	71
5.4.1.4	BBC	72
5.4.2	Experimental Setup	72
5.4.3	Evaluation Metrics	74
5.4.4	Clustering Results	75
5.4.5	Convergence Analysis	84

5.4.6	Different Forms of Distance Fusions	86
5.4.7	Distance Matrix Analysis	89
5.5	Summary	90
6	Dual Distance Adaptive Multiview Clustering	91
6.1	Background and Motivation	92
6.2	Proposed Method	93
6.2.1	Dual Distance	93
6.2.1.1	ELM Embedding Space	94
6.2.1.2	Combination of distance metrics	96
6.2.2	Dual Distance Adaptive Multiview Clustering	96
6.2.3	Optimization	98
6.2.3.1	Update w and F	99
6.2.3.2	Update A	99
6.2.3.3	Parameter Determination	101
6.2.3.4	Stopping Criterion	102
6.3	Experiments	104
6.3.1	Datasets	104
6.3.2	Algorithms and Settings	104
6.3.3	Evaluation Metrics	105
6.3.4	Comparison with Existing methods	107
6.3.5	Performance on Single-view Datasets	111
6.3.6	Combination function of distance matrices	111
6.3.7	Combination of distances with unequal weights	112
6.3.8	Analyzing the parameters	114
6.3.9	Ablation study	118
6.4	Summary	118
7	Conclusions and Future Work	119
7.1	Conclusions	120
7.2	Future Works	121
	List of Author's Publications	123

List of Figures

3.1	Illustration of Octree Structure	27
3.2	Illustration of Feature Extraction	28
3.3	Structure of OCA-ELM with three types of input data	30
4.1	Performance w.r.t. dataset attributes	48
4.2	Influence of trade-off parameters in UFS-ELM	54
4.3	Influence of number of hidden neurons in UFS-ELM	56
5.1	Illustration of normalized distance. When measured by the Euclidean distances, x_1 and x_3 are equally far from x_2 (i.e., $d_{12} = d_{23}$). x_2 is the closest neighbor for x_3 and is further than most neighbors for x_2 , this difference is not expressed by Euclidean distance. When measuring the neighbors of x_2 by normalized distance, x_3 is closer than x_1 (i.e., $\bar{d}_{12} > \bar{d}_{23}$).	63
5.2	Samples of Handwritten Digits	69
5.3	Samples of MSRCv1	69
5.4	Samples of COIL20	70
5.5	Distance matrices of HD	81
5.6	Distance matrices of MSRCv1	82
5.7	Distance matrices of COIL20	83
5.8	Convergence Curves of Loss Function	85
6.1	λ is updated in each iteration until the c -th smallest eigenvalue of the Laplacian vanished	103
6.2	Samples of Handwritten Digits Dataset	104
6.3	Samples of MSRCv1 Dataset	108
6.4	Samples of COIL20 Dataset	108
6.5	Performance of weighted combination of the distances	113
6.6	Influence of the dimensionality of embedded space	115
6.7	Influence of the trade-off parameter	116

List of Tables

3.1	Statistics of ModelNet	32
3.2	Comparing OCA-ELM with the State-of-the-art Shape Classification Algorithms on ModelNet	33
3.3	Comparison of CAE-ELM and OCA-ELM with different input	36
4.1	Specifications of the Benchmark Datasets	43
4.2	Clustering accuracy (%) on benchmark datasets	49
4.3	NMI (%) on benchmark datasets	50
4.4	Purity (%) on benchmark datasets	51
4.5	Running time on the benchmark datasets	52
4.6	Clustering accuracy of UFS-ELM with and without L2,1 norm	57
4.7	Clustering accuracy of UFS-ELM with L2,1 norm and with L2 norm	58
5.1	Specifications of the Datasets	68
5.2	Clustering Performance (%) on Handwritten Digits	77
5.3	Clustering Performance (%) on MSRCv1	78
5.4	Clustering Performance (%) on COIL20	79
5.5	Clustering Performance (%) on BBC	80
5.6	Clustering Performance (%) of DFAN with different within-view distance fusions	87
5.7	Clustering Performance (%) of DFAN with different between-view distance fusions	88
6.1	Comparison of Multiview Clustering Algorithms (%)	109
6.2	Comparison of Combination Functions (%)	110
6.3	Single-view Clustering Performance (%)	111
6.4	Ablation Study(%)	117

Symbols and Acronyms

Symbols

$\mathcal{R}^n$	the n -dimensional Euclidean space
$\ \cdot\ _2$	the L2-norm of a vector or matrix in Euclidean space
$\ \cdot\ _G$	the induced norm of a vector in G-space
$\ \cdot\ _E$	the induced norm of a vector or matrix in probabilistic space
$\odot$	the Hadamard (component-wise) product
$\otimes$	the Kronecker product
$\langle \cdot, \cdot \rangle$	the inner product of two vectors
$\circ$	the composition of functions
∇f	the gradient vector
$\mathcal{C}^k$	the function with continuous partial derivatives up to k orders
$x_{i,k}$	the i -th component of a vector x at time k
$\bar{x}$	the vector with the average of all components of x as each element
$\mathbf{1}$	all-ones column vector with proper dimension
$\mathcal{C}$	the average space, i.e., $\text{span}\{\mathbf{1}\}$
$\mathcal{C}^\perp$	the disagreement space, i.e., $\text{span}^\perp\{\mathbf{1}\}$
$\Pi_\parallel$	the projection matrix to the average space $\mathcal{C}$
$\Pi_\perp$	the projection matrix to the disagreement space $\mathcal{C}^\perp$
$O(\cdot)$	order of magnitude or ergodic convergence rate (running average)
$o(\cdot)$	non-ergodic convergence rate
$\mathcal{N}_i$	the index set of the neighbors of agent i

Acronyms

ADMM	Alternating Direction Method of Multipliers
ALC	Adaptive Locality-constrained Clustering
CAN	Clustering with Adaptive Neighbors
CV	Computer Vision
CLR	Constrained Laplacian Rank
ELM-AE	Extreme Learning Machine Auto-Encoder
ELM-ALC	Adaptive Locality-constrained Clustering with ELM
ELM	Extreme Learning Machine
$ELM_{c^{LDA}}$	Extreme Learning Machine Clustering based on Linear Discriminant Analysis
ELM-CLR	clustering based on Extreme Learning Machine and Constrained Laplacian Rank
ELM-JEC	Extreme Learning Machine for Joint Embedding and Clustering
HOG	Histogram Oriented Gradients
LASSO	Least Absolute Shrinkage and Selection Operator
ML	Machine Learning
NCut	Normalized Cut
PCA	Principal Component Analysis
PCAN	Projected Clustering with Adaptive Neighbors
SC	Spectral Clustering
SLFNs	Single hidden Layer Feedforward neural Networks
SVM	Support Vector Machine
US-ELM	Unsupervised Extreme Learning Machine

Chapter 1

Introduction

1.1 Background

Machine learning algorithms attract much attention over the past decade due to artificial neural networks showing significant performance in diverse applications ranging from image classification, object detection, speech recognition, function approximation, and even video games. Artificial neural networks, usually known as neural networks, are systems consisting of directional linked nodes with parameters, and the values of the parameters can be derived via learning from data observations. Training algorithms are carefully designed to guide the learning of the parameters as an optimization process. Various neural networks and corresponding training algorithms have been reported for machine learning tasks in all three paradigms: supervised learning, unsupervised learning, and reinforcement learning.

Among different unsupervised learning tasks, clustering is the most common and essential one. Clustering is the process that groups the data observations based on their similarity. The clustering results can be directly used for gene expression analysis, image retrieval, and image segmentation. On the other hand, clustering can support many other supervised learning tasks and unsupervised learning tasks (e.g., feature learning, data visualization, and anomaly detection etc.). Consequently, clustering algorithms often act as a component of other machine learning algorithms. Therefore, clustering is also applied in recommender systems, anomaly detection, dimension reduction, and natural language processing.

Without the annotations, clustering mostly relies on the similarities between data observations. The similarity could be artificial, varied with scenarios, and sometimes ambiguous. Generally, the similarity is directly measured from the observations according to a predefined similarity metric. However, the observations may contain noise and redundant features. One way to solve this is by transferring the original observed features to the enhanced representation, namely embeddings. Since neural networks have been used for function approximation, it can generate the embeddings by learning the transfer function in an unsupervised manner.

The learning capability of neural networks grows as the model complexity increases. The learning capability could be superabundant and neural networks may learn an overcomplex function specifically. Overcomplex could increase the variance of the model and lead to poor generalization performance. The tradeoff would indicate that variance increases with the complexity of the neural networks. Regularization embedded in the optimization process aims to reduce the variance and increase the generalization performance of neural networks. Several regularization terms, including group lasso or group regularization, are used to address this problem.

In some real-world applications, the characteristics of a data instance are observed and recorded in different formats. The data with multiple sets of features is called multiview data. An extended topic of clustering focused on multiview data is multiview clustering. The objective is to derive a consistent clustering result from different feature sets. Multiview clustering algorithms are always involved combining the information from multiview feature sets, namely multiview fusion. The information to be fused could be the features, similarities, or clustering results, etc.

Similar to multiview clustering, multiview classification is the multiview version of classification problems. For example, 3D shapes have very different representations like point clouds and voxel grids. Point clouds and voxel grids have their unique structure and require different processing. Thus, 3D shape classification is a multiview classification problem. On the other hand, 3D shapes can be projected to multiple 2D spaces as images. Each image can be considered as a view, and it is again multiview classification.

1.2 Contributions

The previous section briefly reviews the research background of neural networks and multiview machine learning tasks. The objectives of this thesis are to apply neural networks to improve performance of classification and clustering. This section summarizes the main contributions of this thesis.

- This thesis proposes octree based convolutional autoencoder extreme learning machine (OCA-ELM) for 3D shape classification. The proposed approach adopts the octree structure and manipulates several 3D representations to extract high-level features. The extracted feature sets are concatenated and fused by Extreme Learning Machine Auto-Encoder (ELM-AE) to preserve the reconstruction capability. Finally, the classification is done by a typical ELM. The effectiveness of the proposed method is verified by large-scale 3D shape datasets.
- This thesis proposes the unsupervised feature selection based extreme learning machine (UFS-ELM) for clustering. The proposed algorithm adopts the unsupervised feature selection method as regularization for the ELM. This thesis equips ELM with both L2,1 norm and manifold regularization and develops an iterative solution for the problem formulated. To be more specific, the L2,1-norm of output weights is embedded in the loss function, and the intrinsic geometry of ELM's output is constrained by manifold regularization. The output function of ELM directly approximates the clustering indicator of the data observations. Evaluations of the proposed method and other existing ELM based clustering method are conducted on various benchmark datasets.
- For multiview clustering problem, this thesis focuses on the most common and critical problem - information fusion. This thesis proposes to fuse the multiview information in the distance level, namely distance fusion. The proposed within-view distance fusion provides a comprehensive view of the relative magnitude of the pairwise distances. The proposed between-view distance fusion avoids the domination of zero in graph fusion and ensures an unified distance measure for different views. A multiview spectral clustering

algorithm based on the distance fusion is proposed in this thesis. Experimental results demonstrate the effectiveness and the convergence behaviour of the proposed methods.

- This thesis proposes a multiview clustering framework, namely dual distance adaptive multiview clustering, which is robust against data with nonlinear structure. The idea of combining embedding distance and original distance is extended to multiview clustering problems for data observations with a nonlinear structure in the original space. Besides using the product as the combination function, this thesis explores various combinations and studies their performance. Extensive experiments on various multiview datasets show that the proposed approach surpasses the state-of-the-art algorithms in the multiview clustering problem. This thesis also analyzes the properties of the proposed method, such as the combination function and the sensitivity to hyperparameters.

1.3 Organization

The remaining chapters of this thesis are organized as follows:

Chapter 2 gives a comprehensive literature review on the following topics: Extreme Learning Machine, unsupervised feature selection, Laplacian rank constraint, multiview clustering, and 3D shape classification. Meanwhile, the prerequisite knowledge of this thesis, i.e., distance measure, is also included.

Chapter 3 investigates OCA-ELM for 3D shape classification. The OCA-ELM adopts octree data and octree convolution to enhance the feature map, while multiview fusion is handled by extreme learning machine autoencoder. The final classification is done by an ELM.

Chapter 4 investigates regularized extreme learning machines for clustering and proposes the UFS-ELM. Unsupervised feature selection methods embed a regularization term in the optimization equation to select better hidden features, the UFS-ELM utilized unsupervised feature selection to improve the regularized extreme learning machines for clustering.

Chapter 5 investigates the multiview clustering problem and proposes a multiview distance measure called distance fusion. Distance fusion provides a unified scale between multiple views, so the distances of views can be fused directly. A multiview clustering algorithm based on distance fusion is proposed called Distance Fusion with Adaptive Neighbour (DFAN).

Chapter 6 introduces an Extreme Learning Machine based multiview clustering approach called Dual Distance Adaptive Multiview Clustering (DAME). DAMC applies extreme learning machine to learn embedding spaces and uses the embedding distance to enhance the distance metric.

Chapter 7 concludes this thesis and discusses the future research directions.

Chapter 2

Literature Review

2.1 Clustering

Clustering [1] is the general task of organizing data observations into some groups, namely clusters, by exploiting the hidden structure of data. The clusters are bounded based on some concepts of similarity or intrinsic characteristics, such that the data samples in the same cluster are alike while the data samples from different clusters are dissimilar. Different from classification, clustering is an unsupervised learning task that separates data observations into homogeneous groups. Therefore, there is no annotation for the data observations, and the process solely depends on the features of the data. Unlike the unsupervised predictive tasks, clustering tasks usually have all the data observations presented at the first place.

The results of clustering can be directly used for applications such as genetic analysis [2, 3], market segmentation [4, 5], and image segmentation [6, 7]. Clustering is also applied in fundamental problems like dimension reduction [8, 9], feature learning [10, 11], and cluster sampling [12, 13]. Thus, clustering often becomes pre-processing for the more complex tasks such as object detection [14], recommender systems [15, 16], and text classification [17–19].

The clustering problem can be formulated in various ways—for example, the availability of the number of clusters categorizes the clustering problems into different classes. In partitioning clustering, the number of clusters is predefined, and all

data observations are partitioned into clusters simultaneously. Hierarchical clustering methods organize the data observations in a tree structure represented as a dendrogram, partitioning versions of the solution can be derived by cutting the tree at the appropriate depths. Some clustering methods group the data observations based on their distribution or local density. The number of clusters can be an arbitrary number. This thesis focuses on partitioning clustering algorithms.

2.1.1 Spectral Clustering

Spectral clustering is a kind of clustering algorithms based on graph theory. Spectral clustering is named for utilizing the spectrum information of Laplacian of the graph. The spectrum information is the eigenvectors and eigenvalues derived by eigendecomposition on the graph Laplacian.

Graph is a straight way to represent the data observations as nodes and their pairwise relationships as edges. Since the edges connecting the nodes have weights, they can represent the similarities between the data observations. For example, the node pair connected by an edge with higher weight is considered that these two nodes are similar to each other. In clustering perspective, the strongly connected nodes should be assigned to the same cluster and each strongly connected component in the graph is considered as a cluster. However, the number of components in the graph can be different to the desired number of clusters as predefined in the partitioning clustering problem. In the extreme case, all the nodes are connected and there is only one component in the graph. To increase the number of components in the graph, some edges need to be cut or some node pairs need to be disconnected.

An important fact is that the number of components in the graph is equal to the number of zero eigenvalues of the graph Laplacian. The value of the i -th smallest eigenvalue is positively correlated to the number of edges that need to be cut to modify the graph to have i components. The success of spectral clustering is mainly based on the fact that it does not make strong assumptions on the form of the clusters.

For a single-view dataset $X \in \mathbb{R}^{n \times o}$, spectral clustering methods construct an affinity matrix A to measure the pairwise similarities. The degree matrix is a

diagonal matrix, in which the diagonal elements are computed as $G_{ii} = \sum_{j=1}^n A_{ij}$. The graph Laplacian is defined as $G - A$ and the normalized graph Laplacian is $L = G^{-\frac{1}{2}}(G - A)G^{-\frac{1}{2}}$. The clustering is done via eigendecomposition of the graph Laplacian, which can be formulated as:

$$\min_{F \in \mathbb{R}^{n \times c}, F^T F = I} \text{tr}(F^T L F) \quad (2.1)$$

where F is formed by the largest c eigenvectors and c is the desired number of clusters. The rows of matrix F are the embedding of the data points, which can be fed into the k-means to obtain the final clustering results.

2.1.2 Constrained Laplacian Rank

Since the affinity matrix A has an arbitrary number of connected components rather than c , the eigenvectors are not yet the cluster indicator. To deliver an affinity matrix A with exact c connected components, constrained Laplacian rank (CLR) [20] adds a rank constraint $\text{rank}(L_A) = n - c$ to the graph Laplacian L based on the following theorem [21]: The multiplicity c of the eigenvalue 0 of the Laplacian matrix L_A (nonnegative) is equal to the number of connected components in the graph with the affinity matrix A . The formulation of CLR is as follow:

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} & \sum_{i,j}^n \|\mathbf{x}_i - \mathbf{x}_j\| a_{ij} + \alpha \|A\|_F^2 \\ \text{s.t.} & \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1, \\ & \quad \text{rank}(L_A) = n - c \end{aligned} \quad (2.2)$$

where $a_{i,j}$ is an element in A , $\mathbf{a}_i$ is the i -th vector in A , $\mathbf{1}$ is a vector of ones. As shown in Eq. 2.2, by adjusting the affinity matrix A , CLR adaptively changes the neighbors of the data samples. With the rank constraint, the resulting affinity matrix A will have exact c connected components. Hence, a final clustering result can be directly obtained from the affinity matrix A via Tarjan's strongly connected components algorithms [22].

2.2 Unsupervised Feature Selection

Nowadays, the machine learning tasks often come with a large-size feature set. For a specific task, some of the features are more relevant to the task while the rest are useless or even hurtful to the task. The data preprocessing that picks up those relevant features is called feature selection. The unsupervised feature selection methods can be categorized into three classes: filter [23–26], wrapper [27, 28] and embedded method [29–32]. The filter method ranking the features solely depend on the statistical measures of the data and regardless of the subsequent learning model. The wrapper methods always work along with some specific models, the performance of the feature subsets measured by the model are used as a criterion to rank the feature subsets. The embedded methods perform feature selection and specific task simultaneously by embedding the feature selection process in the problem formulation. Efficient robust feature selection (ERFS) [23] presents a supervised feature selection method which jointly minimizes the L2,1 norm on both loss function and regularization. Such formulation results in a weight matrix that is robust to the outliers of the data samples, and of which many rows shrink to zeros to achieve feature selection. Unsupervised discriminative feature selection (UDFS) [29] exploits scatter matrices and feature correlations simultaneously. The learned weight matrix is constrained by L2,1 norm and the features are selected according to the corresponding weight values. Similar to UDFS, Nonnegative discriminative feature selection (NDFS) [30] also adopts L2,1 norm regularization. The key difference between NDFS and UDFS is that NDFS proposes to learn a cluster indicator matrix with nonnegative constraint. Robust unsupervised feature selection (RUFS) [31] employs orthogonal nonnegative matrix factorization to learn the pseudo labels and cluster centroids, which also selects features via L2,1 norms minimization. There are several existing ELM methods that adopt L2,1 norm regularization for supervised learning tasks [33–36], most of which minimize the L2,1 norm of the output weight of ELM to select the best hidden neurons. Recently, autoencoder based deep extreme learning machine (ADELM) [37] replaces random weight generation with autoencoder to improve the fingerprint classification performance, which also employs L2,1 norm to regularize the output weights.

Efficient and robust feature selection (ERFS) [23] introduces a supervised feature selection method which employs L2,1 norm minimization on both the loss function and regularization. It replaces the L1 norm in LASSO (least absolute shrinkage

and selection operator) regression for classification with L2,1 norm of the projection matrix. The optimization problem becomes:

$$\min_W \frac{1}{\gamma} \|X^T W - Y\|_{2,1} + \|W\|_{2,1} \quad (2.3)$$

where W is the projection matrix. As stated in the paper, all the regression coefficients corresponding to a single feature will be penalized as a whole to achieve feature selection. Although both L0 and L2,1 have the effects of feature selection, the latter is convex and can be easily optimized.

Unsupervised discriminative feature selection (UDFS) [29] defines a discriminative score of the weight matrix to determine the discriminative ability of the weighted features, which is incorporated with L2,1 norm of the weight to select the most discriminative features. Similar to UDFS, nonnegative discriminative feature selection (NDFS) [30] also adopts L2,1 norm regularization to select features:

$$\begin{aligned} \min_{F, W} \quad & \text{Tr}[F^T L F + \alpha(\|X^T W - F\|_F^2 + \beta\|W\|_{2,1})] \\ \text{s.t.} \quad & F^T F = I_c, F \geq 0 \end{aligned} \quad (2.4)$$

where $\mathbf{F}$ is the cluster indicator matrix and $\mathbf{L}$ is the Laplacian matrix. The key difference between NDFS and UDFS is that NDFS proposes to learn a cluster indicator matrix $\mathbf{F}$ with nonnegative constraint Such that it can find the clusters and select the features simultaneously. By adding the orthogonal constraints, there is only one nonzero element in each row of $\mathbf{F}$. Robust unsupervised feature selection (RUFS) [31] employs orthogonal nonnegative matrix factorization to learn the pseudo labels and cluster centroids. It also selects features via L2,1 norms minimization.

2.3 Extreme Learning Machine

Neural networks have become the most popular tools in machine learning, which effectively design the basis of embedding space applicable to specific tasks. Extreme learning machine (ELM) [38] is a kind of single hidden layer feedforward neural networks famous for its short training time and universal approximation capability. The key idea of ELM is tuning the output weights with fixed hidden neurons to

ensure universal approximation capability and optimizing the model only once (i.e., without iterative training). ELM can achieve nonlinear mapping from the original space to embedding space using the weighted sum of nonlinear activation functions. During the training, the hidden layer is not learned, but the weights of the output neurons are obtained by solving the optimization problem formulated by some learning criteria and regularizations. ELM has been adopted to solve both supervised learning problems [39, 40] and unsupervised learning problems [41–44]. The applications of ELM include, but are not limited to, feature learning [45–48], dimension reduction [49], 3D shape analysis [40, 50], traffic sign recognition [51], bioinformatics[52], and market research [53].

The original ELM has two layers of parameters: the hidden layer and the output layer. The two layers are corresponding to the two phases of the learning process: random feature mappings and output weights learning. The hidden layer contains the weights and the bias of hidden neurons, with which a nonlinear piecewise continuous activation function maps the data from the original input space to a hidden space. The output layer contains the output weights that transfer the hidden neurons to the output of the ELM to approximate the expected data labels. The parameters of the hidden layer are randomly initialized and fixed. In other words, they are non-learnable parameters. The output weights are learned only once and other hyparameters such as the width of layers and the activation function are predefined.

For each single hidden neuron, the weights are initialized as $\mathbf{a} \in \mathbb{R}^{n_f}$ and the bias is a constant $b \in \mathbb{R}$, where n_f is the number of input features. Given data $X = [\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_{n_i}^T]^T$, where n_i is the number of input data observations. Each input data observation is normalized as $\mathbf{x}_i \in [-1, 1]^{n_f}$. The output of each hidden neuron is computed as:

$$h_i(\mathbf{x}) = g(\mathbf{a}_i, b_i, \mathbf{x}) \quad (2.5)$$

where $g(\cdot)$ is a nonlinear piecewise continuous activation function which is predefined as [54]:

- Sigmoid function

$$g(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b))} \quad (2.6)$$

- Hard-limit function

$$g(\mathbf{a}, b, \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a} \cdot \mathbf{x} - b \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.7)$$

- Gaussian function

$$g(\mathbf{a}, b, \mathbf{x}) = \exp(-b\|\mathbf{x} - \mathbf{a}\|^2) \quad (2.8)$$

- Multiquadric function

$$g(\mathbf{a}, b, \mathbf{x}) = \sqrt{\|\mathbf{x} - \mathbf{a}\|^2 + b^2} \quad (2.9)$$

and the output matrix of hidden layer is as:

$$H = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_1(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_m(\mathbf{x}_n) & \cdots & h_m(\mathbf{x}_n) \end{bmatrix} \quad (2.10)$$

where m is the number of hidden neurons. The learning process of ELM is mainly to adjust the output weights to minimize both the norm of the output weights and the error between the target output and the actual output of ELM. In the original implementation of ELM [55], the output weights matrix is computed by using the minimal norm least square method:

$$\boldsymbol{\beta} = H^\dagger T \quad (2.11)$$

where $\boldsymbol{\beta}$ and T are respectively the output weights and the target output, and $H^\dagger$ is the Moore-Penrose generalized inverse of matrix H [56].

Without considering the structural risk, the original ELM can be seen as empirical risk minimization algorithm which tends to produce trivial solutions. To address this problem, regularized extreme learning machine [57] minimizes the sum error square of resident between target value and output along with the Frobenius norm of the output weights. Optimally pruned extreme learning machine(OP-ELM) [58] uses multiresponse sparse regression(MRSR) [59] to rank the hidden neurons, which can be considered as L1 regularization. TROP-ELM [60] extends OP-ELM [58] with Tikhonov-regularized prediction sum of squares (PRESS) [61] to select the best hidden neurons, which is a solution with both L1 and L2 regularization.

Convolutional Auto-encoder Extreme Learning Machine (CAE-ELM) is a novel learning framework designed for 3D shape classification, 3D shape retrieval and 3D shape completion. In CAE-ELM, two types of data are used to obtain both global and local features. The input data first goes through a convolution layer with random weight convolution kernel, then followed by a pooling layer to maintain the rotation invariance of the features. The extracted features are compressed by an ELM-AE and distinguished by an ELM classifier.

2.3.1 Extreme Learning Machine for Clustering

ELM is first designed for solving supervised learning problem e.g. classification and regression [62, 63]. It is then extended to unsupervised learning problems [64–66] mainly the clustering problems.

US-ELM [65] is proposed as a clustering algorithm that uses graph Laplacian to learn the embeddings of the data observations, whose output is restricted to be an orthogonal matrix. To preserve the intrinsic geometry, manifold regularization is adopted for the embedding and the output weights matrix is regularized by Frobenius norm.

$$\begin{aligned} \min_{\beta} \quad & ||\beta||_F^2 + \lambda \text{Tr}(\beta^T H^T L H \beta) \\ \text{s.t.} \quad & (H\beta)^T H\beta = I \end{aligned} \tag{2.12}$$

where L is the Laplacian matrix of the input data observations. To solve the optimization problem with hard constraint, US-ELM computes the eigendecomposition of the hidden neurons and learns the output weights by stacking the eigenvectors corresponding to the smallest eigenvalues. The clustering result is derived by running k-means on the output, which is an orthogonal matrix.

Similar to US-ELM, ELMc^{LDA} [66] initially runs k-means on the hidden space and labels the dataset with the cluster indicator. The output weights are then obtained by supervised LDA on the pseudo-labeled data observations, and the clusters are found by running k-means on the output vectors.

US-ELM and ELMc^{LDA} split the clustering problem into parts, and the solutions rely on other algorithms. Contrast with that, ELM-JEC [64] is an end to end solution for the clustering problem, which learns the embeddings and finds the clusters simultaneously. ELM-JEC merges the clustering process into the optimization by

adding the cluster centroids into the formulation. The objective function of ELM-JEC is as follow:

$$\begin{aligned} \max_{\beta, G, F} \quad & \text{Tr}(\beta^T \bar{H}^T \bar{H} \beta) - \lambda \|\bar{H} \beta - FG^T\|_F^2 - \gamma \text{Tr}(\beta^T \bar{H}^T L \bar{H} \beta) \\ \text{s.t.} \quad & \beta^T \beta = I_n \end{aligned} \quad (2.13)$$

where G is the cluster centroid matrix and F is the cluster indicator matrix. The output matrix of the hidden layer is replaced by the centered version, which is computed as $\bar{H} = (I_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T) \mathbf{H}$. In this thesis, to achieve best performance, we follow ELM-JEC to use the centered hidden layer output $\bar{H}$. The output weights matrix is obtained by eigendecomposition to satisfy the hard constraint. The cluster of each data point is then indicated by its closest cluster centroids.

ELM-CLR [67] leverages ELM to improve the performance of adaptive graph learning, whose mathematical formulation is as follows:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \sum_{i,j=1}^n (\bar{d}_{ij} \bar{p}_{ij} w_{ij} + \gamma w_{ij}^2) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \mathbf{w}_i^T \mathbf{1}_n = 1 \\ & \text{rank}(L_S) = n - c \end{aligned} \quad (2.14)$$

where $\bar{d}_{ij}$ and $\bar{p}_{ij}$ are the pairwise distances in original space and embedding space, and w is the pairwise similarity. Specifically, ELM-CLR initializes the similarity matrix W based on the pairwise distances measured in the original space. The ELM-based embedding space is learned from the similarity matrix. Then, the similarity matrix is adaptively tuned based on the product of both distances under the Laplacian rank constraint. Once the Laplacian rank constraint is satisfied, the optimal solution is obtained as a graph with its number of components equaling to the desired number of clusters.

2.4 Multiview Clustering

In real-world applications, the data observations can be recorded in multiple formats by multiple sensors. For example, autonomous vehicles equipped with LiDAR and cameras can record the vision as images or point clouds. The raw data,

along with the features extracted from it, can also be regarded as multiview data. For example, various kinds of handcrafted features can be generated from an image: scale-invariant feature transform (SIFT) [68], histogram of oriented gradient (HOG) [69], and local binary pattern (LBP) [70]. A 3D object can be described by the signed distance functions, occupancies, and the averaged normal vectors of the points or voxels that are made of [40]. Such data with multiple feature sets are referred to as multiview data. Each feature set is referred to as a view of the data observations.

Due to the different representations of the features, different views may require different processing. Of course, high-level features can be extracted from multiple views and concatenated to form a single view of features [71]. However, each view has its individual unique statistical information. Simple concatenation ignores the correlation of the features within the same view. The results of the experiments [72] confirm the above arguments. Therefore, the learning of multiview data is an attractive and promising research field. On the other hand, The concatenated single view data is contradictory to those processes that extract local features. Thus, the multiple views usually are handled separately as regular data (i.e., single view data), which generates multiple results for the same single task. The multiple results can be more or less inconsistent, and most tasks request a single answer. The view-specific features are used to optimize a global function, reduce the influence of imprecision and noise from every single view, as well exhaust the otiose views. Hence, the critical point of solving tasks on multiview data is to combine the multiview feature to generate a single result, which is referred to as information fusion.

The tasks on multiview data include multiview clustering, multiview classification [73], and regression [74]. Clustering [42, 43] is a kind of unsupervised learning tasks [41, 44] in computer vision and pattern recognition. Clustering intends to split the data into subsets so that the instances within the same subset are similar, and they are dissimilar to the instances of other subsets. It also explores the statistical structure of the data. Multiview clustering has the same objective as single-view clustering, except the data has multiple sets of features.

There are several kinds of existing multiview clustering methods, such as multi-kernel multiview clustering [75–77], multiview subspace clustering [78–83], multiview clustering with nonnegative matrix factorization [84–87], and multiview spectral clustering [88–91]. Multi-kernel multiview clustering designs kernel matrices for multiple views. A global kernel can be obtained from the linear or nonlinear combination of the kernels. The global kernel is used in kernel k-means [75, 92] or spectral clustering methods [76, 77] to produce the clustering results. Subspace clustering [81] reconstructs the data matrix by the product of the data matrix itself and a coefficient matrix with zero diagonal. The coefficient matrix reveals how a data sample can be represented by the rest data samples, which indicates the similarity between the samples. Thus, a similarity matrix can be constructed from the coefficient matrix. The final clustering results can be computed by feeding the similarity matrix to a spectral clustering method. Similar to subspace clustering, nonnegative matrix factorization clustering [93] reconstructs the original data matrix by the product of a basis matrix and a coefficient matrix. The multiview versions [84, 94] of the above two kinds of clustering methods minimize the differences between the coefficient matrices from different views to find a global coefficient matrix for spectral clustering methods [95, 96]. The spectral clustering method constructs the affinity matrix of the data and finds the clusters by computing the eigenvectors of the Laplacian of the affinity matrix. Multiview spectral clustering methods unify the graphs into a global graph. While a consistent clustering result for multiple views can be acquired from the eigendecomposition of the global Laplacian matrix. Specifically, auto-weighted multiple graph learning (AMGL) [88] constructs graph and computes Laplacian of the graph for each view. The global Laplacian is defined as the weighted sum of the Laplacian matrices. Parameter-weighted multiview clustering (PwMC) and self-weighted multiview clustering (SwMC) [89] learn a global graph by approximating the graphs from all the views. Graph learning for multiview clustering (MVGL) [90] computes the weighted sum of graphs as the global graph. Graph-based multi-view clustering (GMC) [97] optimizes a global graph by minimizing the difference between it and all view-specific graphs. The view-specific graphs are also iteratively optimized based on the global graph. Multi-graph fusion for multi-view spectral clustering (GFSC) [98] computes view-specific representation coefficient matrices based on the self-expressiveness property. Since the coefficient matrices behave like the similarity matrices, a global graph is learned by approximating all coefficient

matrices. Cross-view matching clustering (COMIC) [99] learns a representation by reconstruction, computes similarity matrix for each view, and then learns a connection graph wherein the data samples of the same class are connected for each view. The cluster assignment consistency in COMIC is achieved by minimizing the difference between each of the two connection graphs.

Under this condition, the initial KNN graphs in GSF are usually constructed with a large k to reduce the number of the disconnected pairs. This increases the redundant connections and reduces the locality of the data. Inspired by this, we exploit a new way to fuse the structure information from multiple views in distance level namely distance fusion. Specifically, we use the mean of the distance matrices as the global distance matrix. To ensure a unified scale of the distances, the distance matrices are normalized before the between-view distance fusion. Row normalization and column normalization result in two different matrices and provide two perspectives of each pairwise distance. We propose within-view distance fusion to fuse these two matrices to include the metrics of both perspectives. In this way, each pairwise distance is compared with all the other pairwise distances related to the data pair. Unlike multi-kernel methods that only consider the data pair, our measure of similarity is obtained by taking into account the data pair and their neighbors.

Most recently, the inCOMplete muLti-view cluStEring via conTrastivE pRediction (COMPLETER) [100] aims at missing views, it solves multiview fusion and data completing problem simultaneously. Specifically, COMPLETER minimizes the mutual information to construct a consensus representation of multiple views and minimize the conditional entropy of between views to complete missing views. Another work aims at incomplete multiview clustering is the generalized IMC with flexible locality structure diffusion (GIMC_FLSD) [101], which is a graph-regularized matrix factorization model. GIMC_FLSD fuses individual view into a unified term without introducing any extra penalty parameter. The negative impact of incomplete views can be minimized by adaptively determining the importance of views. Consensus Multi-view Clustering (CMC) [102] is a nonnegative matrix factorization (NMF) multiview clustering method, which adds regularization for NMF and learns a more robust NMF with the aid of binary relationship matrix.

2.4.1 Multiview Spectral Clustering

Spectral clustering is a conventional clustering technique, which constructs an undirected graph of the data objects and reveals the data structure via eigen-decomposition of the graph Laplacian. In multiview spectral clustering, with the excellent clustering capability of spectral clustering, the remaining and most important work is to learn the common graph or eigenvectors consistent among all the views. The distinction between different multiview spectral clustering methods is the way of multiview information fusion. Co-regularized multiview spectral clustering (CMSC) [103] is an early work of multiview clustering. For each view, CMSC computes the eigenvectors of the graph Laplacian and forms a cluster indicator matrix with them. By minimizing the L2 norm of their product, it then regularizes a common centroid cluster indicator (i.e., a consensus set of eigenvectors) with each view-specific cluster indicator. Multiview clustering with adaptive neighbours (MLAN) [104] measures the pairwise distances from all the views and constructs the common graph based on the weighted sum of the distance matrices. MLAN then adjusts the weights and the graph simultaneously to gain the result with desire number of clusters. Auto-weighted multiple graph learning (AMGL) [88] constructs graph and computes graph Laplacian for each view. The information fusion is done by combining all the Laplacian matrices. Graph learning for multiview clustering (MVGL) [90] defines the common graph as the weighted sum of the view-specific graphs. Parameter-weighted multiview clustering (PwMC) and self-weighted multiview clustering (SwMC) [89] compute the differences between the common graph and each of the view-specific graphs, then minimizes the sum of the differences by rescaling them. Graph structure fusion (GSF) [105] fuses multiple graphs into a common graph via Hadamard product. GSF learns a global graph of the multiview data. Specifically, GSF adopts the Hadamard product of the affinity matrices as the global affinity matrix. However, Hadamard product could remove the worthwhile edges while removing the redundant edges in the graphs. For example, two data samples are connected in most of the graphs but a single one. Regardless of how strong they are connected, the data pair will be disconnected during graph fusion.

The performance of multiview spectral clustering methods primarily relies on the graphs that generally constructed based on the pairwise distances in the original feature spaces. In an ideal space, all the nearest neighbours of a data object are in the same cluster. The gap between clusters must be greater than the largest

pairwise distance in the clusters. Thus, both the pairwise distances and the resulting graph reflect the actual pairwise relationships, from which the best clustering performance can be derived. However, in practice, the original space where the data lie on may not be homeomorphic to the Euclidean space. Therefore, the Euclidean distance measured in these spaces cannot accurately describe the similarity between data objects.

2.4.2 Multiview Learning with Adaptive Neighbors

For multiview dataset, the clustering results optimized for different views could be inconsistent. A consistent clustering result can be obtained from a similarity matrix shared among all the views. MLAN [104] first measures the squared Euclidean distances for each data pair, then computes the common similarity matrix based on the weighted sum of view-specific pairwise distances. The optimization of the similarity matrix and the weights of distances are expressed as following formulation:

$$\begin{aligned}
& \min_{A \in \mathbb{R}^{n \times n}} \sum_v \sqrt{\sum_{i,j}^n \|\mathbf{x}_i^v - \mathbf{x}_j^v\|_2^2 a_{ij}} + \alpha \|A\|_F^2 \\
& \text{s.t.} \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1, \\
& \quad \text{rank}(L_A) = n - c
\end{aligned} \tag{2.15}$$

where A is the global similarity matrix, and L_A is the Laplacian of A . The weights of views w is implicitly defined in the formulation, which can be optimized based on S as follow:

$$w_v = \frac{1}{2\sqrt{\sum_{i,j} \|x_i^v - x_j^v\|_2^2 s_{ij}}} \tag{2.16}$$

Therefore, the optimal solution is obtained by iterative updating S and w . MLAN computes the dot product of A and the distances for each view. Minimizing the dot products will assign high similarities for small distances. The fusion of multiview is achieved by the first term in the formula as the linear combination of the dot products. By sharing A across multiple views, consistent clustering results can be obtained. The second term is a regularization term to avoid trivial solutions. The rank constraint of Laplacian controls the number of components in the graph so

that the result has the desired number of clusters. A consistent clustering result is derived from S via Tarjan's strongly connected components algorithm.

2.5 3D Shape Classification

Computer graphics is an important field in computer science which applies computational methods on geometric objects. Applications of computer graphics includes animation [106, 107], movies [108] and other digital arts [109]. A fundamental cue in most of the sub-fields of computer graphics is 3D shapes of the underlying objects, which carry information of surfaces and spaces. 3D shapes have been used in many different practical applications such as archeology [110], video games [111] and mechanical engineering [112]. As a result, many researchers have been focusing on the fields of 3D shape analysis including shape classification, shape segmentation and shape reconstruction.

The representations of 3D shapes could be classified into two categories: projection-based and geometry-based. In projection-based methods [113], the 3D shapes are first rendered as 2D images from multiple views and use the 2D images to represent the 3D shapes. Each 2D image represents a portion of the 3D object. The projection-based methods convert shape analysis problems into image analysis problems by making inferences about the 3D shapes from the 2D images. With the aid of the powerful Convolutional Neural Network (CNN) [113], the projection-based methods are prominent in shape classification and shape retrieval. [114] gives a comprehensive study on the voxel-based CNNs and multiview CNNs for 3D object classification. In MVCNN [115], each view of the 3D shape is passed through the first part of the network separately, aggregated at a view-pooling layer using max-pooling, and then passed through the remaining portion of the network. However, MVCNN requires a complete set of predefined views of the 3D shape, which is unlikely to happen in some scenarios. Similar to MVCNN, RotationNet [116] also uses CNN to deal with multiview images. It can still classify an object using a partial set of multiview images that may be sequentially observed by treating view-labels associated with pictures as latent variables during training. GIFT [117] enhances matching accuracy by ignoring non-discriminative views and preserves the views with large discriminative scores. DeepPan [118] projects the 3D shapes into panoramic views as the representation. Light field descriptor (LFD) [119] considers

the silhouettes of the 3D shapes. Geometry image [120] transforms the general mesh models into planar 2D grids to generate planar parameterization, which contains complete surface information. MVD-ELM [71] employs depth maps to construct the 2.5D images, containing both view information and depth information.

Lately, view-based graph convolutional network (View-GCN) [121] constructs view-graph with the 2D images generated by projection of 3D shape. It utilizes graph convolutional neural network to hierarchically reveal the relationship between multiple views and learns representative features. Relation-shape convolutional neural network (RS-CNN) [122] converts the irregular point cloud problem into regular grid problem by learning the local features from the relationship between geometric priors. Local phase 3D CNN (LP-3DCNN) [123] presents rectified local phase volume (ReLPV) block to replace traditional 3D convolution layer saving computational and memory cost. FG3D-Net [124] is another view-based 3D shape classification method leveraging region proposal network to extract common semantic part of different views. The common semantic part is then fused into a global shape representation containing the local features of the 3D objects. Thus, the fine-grained details can be learnt from the common semantic parts. CAR-Net [125] introduces correspondence-aware representation (CAR) module to explore the correspondences both inside and between views. The correspondences contains geometric information of the 3D shapes. With relationship information of corresponding viewpoint, better feature representation can be learnt from the correspondence pairs. The CAR module can then be added into conventional CNN backbone for 3D shape classification showing impressive performance.

Chapter 3

Octree based Convolutional Autoencoder Extreme Learning Machine for 3D Shape Classification

Chapter 3 proposes Octree based Convolutional Autoencoder Extreme Learning Machine for 3D shape classification. Section 3.2 describes the training and inference of the proposed method. Section 3.3 presents the experimental results to demonstrate the performance of the proposed algorithm. Section 3.4 summarizes this chapter.

3.1 Background and Motivation

The increasing number of 3D contents raises the demand for shape classification algorithms to deal with the large-scale shape repositories [126], as it is essential for retrieving, rearranging, and browsing the shape repositories. Most state-of-the-art methods designed for 3D shape classification begin by extracting features from the input representation of the 3D shapes that contain the most useful information, then proceed to practical problems based on the extracted features. Therefore, a carefully designed representation of 3D shape is the key to successful classification methods.

Projection-based methods have been dominant in shape classification for a long time. Although they are less sensitive to the noisy meshes or surfaces, they may lose crucial spatial information of the 3D shape. On the other hand, geometry-based methods retain the complete information of the 3D shapes. Such methods utilize geometric information, e.g., volumetric occupancy grids, point sets, normal vectors of surfaces, etc., as direct input to the processing pipeline. When the convolutional operation is extended to 3D data using 3D convolution, the geometry-based methods become superior to the projection-based methods. 3DShapeNets [126] utilizes the voxel representations of 3D shapes and trains a 3D convolutional neural network to extract features. VoxNet [127] applies binary voxel grids as input and trains 3D CNN for feature extraction. PointNet [128] converts 3D shapes into point clouds and trains CNN with the mesh density information. Edge convolution network (ECC) [129] uses filter weights conditioned on the edge labels, extending the convolution operator from regular grids to arbitrary graphs dynamically for each input shape. Spherical Harmonic (SPH) [130] obtains a rotation-invariant representation by calculating the restriction of storing the norm of each frequency and the function of concentric spheres. Octree-based Convolutional Neural Network (O-CNN) [131] builds octree convolution upon the octree representation of 3D shapes. Since O-CNN restricts the operations on a sparse set of occupied octants, such a method applies to high-resolution 3D shapes without losing details. Instead of a single input representation, Convolutional Auto-Encoder Extreme Learning Machine (CAE-ELM) [50] combines the binary occupancy voxel grids with the signed distance field (SDF) data to describe both local and global structure. The

geometry-based methods have recently outperformed the projection-based techniques on the public large-scale 3D model datasets and become more potent with rapid hardware development.

We propose octree-based convolutional autoencoder extreme learning machine (OCA-ELM) with octree convolution for feature extraction and Extreme Learning Machine (ELM) [132] for classification. We further equips OCA-ELM with voxel and SDF input data. The features extracted by ELM-AE [45] can then avoid information loss and data redundancy. The contributions of this chapter are as follow:

- We propose OCA-ELM combining octree convolution and extreme learning machines for 3D shape classification.
- We conduct experiments for OCA-ELM with only octree branch and show that the idea of random neuron in ELM can be applied to octree convolution filter. The random initialized and fixed convolution filter extracted features containing details of the shape, based on which the output layer of ELM can better distinguish the shapes.
- We provide experimental results demonstrating the efficiency of the proposed OCA-ELM.

3.2 Methodology

This section describes the architecture of the proposed OCA-ELM in detail and shows how to train the OCA-ELM for 3D shape classification. The OCA-ELM consists of three stages. The first stage is feature map generation, mainly being comprised of a convolution layer and a pooling layer. The next stage is, in essence, ELM-AE. The ELM-AE is initialized in the equal dimension representation, where the calculated output weights replace the random input weights. Finally, a typical ELM classifier is adopted for 3D shape classification based on the extracted features.

3.2.1 Input Representations

The proposed OCA-ELM has three input branches with different 3D representations: voxel grids, signed distance fields (SDF), and octree contains normal vector information. The feature extraction for the octree input is different from the other two inputs, which requires octree convolution and octree pooling. A n^3 voxel cost $O(n^3)$ memory in conventional voxel representation. The octant information is saved on a 1-D list and the neighboring information of the octants is stored in a hash table. The memory cost is reduced to $O(n^2)$. Based on the hash table, the neighboring octant of a center octant can be found in constant time. Therefore, the computation cost of octree is only $O(n^2)$. While for conventional voxel convolution, the computation cost is $O(n^3)$. In addition, only non-empty octant is stored in the octree and involved in the octree operation. The computation overhead is further reduced. Hence, by applying octree, the memory cost and computational cost can be saved.

The voxel data is a boolean tensor. Each cell in the tensor stores a binary number to indicate whether the 3D shape mesh occupies the cell. A cell will be assigned 0 if it is empty; otherwise, the cell's value is 1. The SDF data is obtained by sampling the distance of each cell to its closest surface of the 3D shape polygonal. A positive value will be assigned to the cell outside the polygonal, while the cells inside the shape surface have a negative value. Both of these two types of 3D representation are commonly used in the field of computer graphics. However, the voxel grids used in CAE-ELM have no orientation information, and the SDF data only has two directions. The orientation information makes the feature maps robust. It is useful in distinguishing the fine-grained 3D shapes.

To this end, the proposed OCA-ELM exploits the octree with averaged normal vectors in OCA-ELM. First, each input 3D shape is scaled into a cube with unit size and divided into 8^d octants based on predefined depth d of the octree. An illustration of these process is shown in Figure 3.1. The 3D shape is then converted into points sampled from the surfaces. The normal vector of these points is computed. Each non-empty octant will cover many points; the average of the normal vectors is taken as the octant's input signal. Moreover, a zero vector will be assigned for the finest octant that contains no surface point.

FIGURE 3.1: Illustration of Octree Structure

3.2.2 Feature Map Extraction

According to the ELM theory [55], the networks constructed with random generated kernels weights are more generalizable. Moreover, the universal approximation capability of ELM can be achieved by using any nonlinear piecewise continuous function. To generate the feature map, OCA-ELM first applies octree convolution to the input octree data.

Unlike the octree convolution in the O-CNN trained with backpropagation, the convolution kernel of octree convolution in OCA-ELM is randomly initialized and fixed through the end. Bias terms are added to the octree convolution layer. Besides, the batch normalization layers and scale layers are removed. Similar to O-CNN, OCA-ELM has a max-pooling layer after the convolution layer to cope with the shape variance. With different weights, multiple feature maps can be generated.

3.2.3 Feature Extraction

FIGURE 3.2: Illustration of Feature Extraction

Each ELM-AE is merely trained for its corresponding feature map generated with unchanged weights during the training phase.

Given pooled feature maps of n samples $P = [\mathbf{p}_1, \dots, \mathbf{p}_n]$, the hidden neuron can be computed as:

$$h = g(\mathbf{w}^T \mathbf{p} + b) \quad (3.1)$$

where $\mathbf{w}$ is the orthogonal weights and b is the random bias. The ELM-AE with equal dimension representation is adopted, in which the number of hidden neurons is the same as the number of data samples. So the output matrix of the hidden layer is $H = [\mathbf{h}_1, \dots, \mathbf{h}_n]$. The input feature maps can be set as the target labels shown in Eq.(3.2):

$$H\boldsymbol{\beta} = P \quad (3.2)$$

It can be solved as shown in Eq.(3.3)

$$\boldsymbol{\beta} = H^{-1}P \quad (3.3)$$

Hence, the features can be extracted as

$$E = H\boldsymbol{\beta} \quad (3.4)$$

3.2.4 Classification

Given n input samples and u autoencoders with each autoencoder having l hidden nodes, OCA-ELM concatenates the extracted features from octree data as $E_{octree} = [E_1, \dots, E_K]$ and combines the features from three different representations as $E_{total} = [E_{octree}, E_{voxel}, E_{SDF}]^T$. The structure of OCA-ELM accepted three types of input data is shown in Figure 3.3. Assume each input sample has an extracted feature vector $\mathbf{e} = [e_1, \dots, e_m]$ where $m = 3 \times u \times l$, the output of the ELM classifier is:

$$f_m(\mathbf{e}) = \sum_{i=1}^m \beta_i h_i(\mathbf{e}) = \mathbf{h}\boldsymbol{\beta} \quad (3.5)$$

FIGURE 3.3: Structure of OCA-ELM with three types of input data

where $\mathbf{h}(\mathbf{e}) = [\mathbf{h}_1(\mathbf{e}), \dots, \mathbf{h}_m(\mathbf{e})]$ is the output of hidden layer with respect to input features $\mathbf{e}$; $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m]^T$ is the output weights of the ELM classifier. Given the target labels T of the features E , the ELM classifier is trained by solving:

$$\min_{\boldsymbol{\beta}} \|H\boldsymbol{\beta} - T\|^2 + \frac{1}{C}\|\boldsymbol{\beta}\|^2 \quad (3.6)$$

where H is the output matrix of the hidden layer. The output weights can be obtained as

$$\boldsymbol{\beta}^* = \begin{cases} (H^T H + \frac{I}{C})^{-1} H^T T & M \ll N \\ H^T (H H^T + \frac{I}{C})^{-1} T & otherwise \end{cases} \quad (3.7)$$

After training, with the optimal $\boldsymbol{\beta}^*$, the class label can be predicted as:

$$f_L(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}^* \quad (3.8)$$

3.3 Experiments

This section reveals the classification capability of OCA-ELM by benchmarking its performance against state-of-the-art algorithms for shape classification on Princeton ModelNet [126].

3.3.1 Datasets

The Princeton ModelNet is a comprehensive clean collection of 3D CAD models with 127,915 CAD models and 662 object categories. The clean subsets of Princeton ModelNet are adopted: ModelNet10 and ModelNet40. Although both the subsets have the orientation-aligned version, in order to test the sensitivity to rotation variance, we adopt aligned ModelNet10 and unaligned ModelNet40 with the standard training and testing split shown in Table 3.1.

For both ModelNet10 and ModelNet40, OCA-ELM is setup with the same hyperparameters. Most shapes in ModelNet datasets have many artifacts such as flipped normals, non-manifold structures, and overlapped triangles. To obtain the correct normal information, we follow the original paper of O-CNN [131] when converting the input 3D shape to an octree structure. We first sample the surface points by virtually scanning the model from 14 different perspectives, then the obtained

TABLE 3.1: Statistics of ModelNet

Dataset	Aligned?	No. of Training	No. of Testing
ModelNet10	Yes	3991	908
ModelNet40	No	9843	2468

points will be rotated to produce 12 copies to achieve rotation invariance of the system. As each shape is contained in a cube, the directions of 14 perspectives are from 6 faces and 8 corners of the cube. The 12 rotated copied are generated by rotating the shape along the upright axis by 30 degrees each time. Each of the 12 copies is converted into a 3-depth octree with the resolution of 8^3 . All the 12 octrees are concatenated as an entirety to represent an input model. The weights in the octree convolution layer are of the kernel size equal to 3, and the size of pooling kernel is 2. The number of feature maps and autoencoders are set to 27.

The same hyperparameters are used for both the voxel branch and SDF branch. The convolution kernel and pooling kernel are of sizes 5 and 2, respectively. The three different branches will have the same width in the E_{total} . Therefore, both the voxel and SDF branches have 96 feature maps. The normalization constant for ELM-AE and ELM is set to 10^{-6} and 0.1.

3.3.2 3D Shape Classification

Table 3.2 shows the performance of OCA-ELM on ModelNet10 and ModelNet40. The classification accuracy is the average precision on all the classes. OCA-ELM has superior capability when compared to state-of-the-art algorithms, including hand-crafted methods, projection-based methods, and geometry-based methods.

TABLE 3.2: Comparing OCA-ELM with the State-of-the-art Shape Classification Algorithms on ModelNet

Algorithms	ModelNet40	ModelNet10
SPH [130]	68.23%	79.97%
LFD [119]	75.47%	79.87%
3DShapeNets [126]	77%	83.5%
DeepPano [118]	77.63%	85.45%
ECC [129]	83.2%	90.0%
Geometry Image [120]	83.9%	88.4%
3D-GAN [133]	83.3%	91.0%
VoxNet [127]	83%	92%
MVD-ELM [71]	81.39%	88.99%
O-CNN (3-depth) [131]	87.1%	-
CAE-ELM [50]	84.35%	91.41%
OCA-ELM (ours)	88.04%	92.17%

Aided by the octree representation, OCA-ELM surpasses CAE-ELM on ModelNet40 by 3.69% classification accuracy. This is expected because OCA-ELM predicts the class label with the surface orientation information, which is not considered in CAE-ELM. It can be concluded that surface orientation is vital information in distinguishing 3D shapes. We also compare OCA-ELM to the O-CNN trained by back-propagation. For a fair comparison, the octree structures accepted by OCA-ELM and O-CNN are both of depth 3. Moreover, since each 3D shape is rotated to generate 12 copies, O-CNN can summarize the output layer for each copy by using the voting approach [127]. The result of OCA-ELM is slightly better than the result of O-CNN reported in [131] as shown in Table 3.2.

It can be seen that other algorithms are comparable to OCA-ELM on ModelNet10, but the performance gap is larger on ModelNet40. The orientations of the 3D CAD models in ModelNet10 are manually aligned by human beings, while the ModelNet40 released by Princeton is not aligned. Thus, it can be concluded that OCA-ELM has a preponderance in recognition of unaligned 3D shapes. The reason is, for each 3D shape, the octrees are generated from different poses of the model. The proposed OCA-ELM combines all the poses into a single input represent the 3D shape. Such design is naturally insensitive to the orientation variance.

The OCA-ELM variants with different input representations are tested on ModelNet to explore the impact of the octree input representation. The results are shown in Table 3.3. Since the different input branches have different feature width, we adjust the number of feature maps, so that feature matrices E_{octree} , E_{voxel} and E_{SDF} have the same size. In this case, the OCA-ELM with octree input is slightly stronger than the previous CAE-ELM with the other two input types. Although CAE-ELM accepted the combination of voxel and SDF is better than OCA-ELM with single octree input, the OCA-ELM with triple input has achieved the best accuracy.

To analysis the computational overhead of proposed method, we run OCA-ELM on a desktop with Intel i5-11600KF CPU and Geforce GTX 1050 Ti graphics card. Since feature extraction via ELM autoencoder has the same computational overhead for different branches, we only compare the time consumption of convolution stage of different branches. For the whole training set of ModelNet10, the data is converted into the three types of data with same resolution $16 * 16 * 16 = 4096$. For

the same training set, SDF branch and voxel branch takes 37.65s and 46.83s respectively, while octree branch with two convolution layer only takes 27.83s. Thus, using octree convolution also saves the time consumption.

OCA-ELM uses octree with depth of 3. Specifically, the input depth3 octrees are converted into depth2 octree by pooling. The depth2 octree is used as input for ELM autoencoder to extract features. To analyze the performance of octree with different depths, we construct the depth4 octrees of ModelNet10. To be fair and ensure the input dimension of the ELM autoencoder, the depth4 octrees are also converted into depth2 octree via two sets of octree convolution and pooling. We evaluate the performance of single-branch octree OCA-ELM with different depths of octree. Compared to depth3 octree, the accuracy decreased from 89.57% to 84.47%. The reason of performance decrements is that the feature map generation via two random convolution layer can lose too much information.

When adopting octree data, the improvement of performance should be more significant on unaligned ModelNet40 than aligned ModelNet10. Because the octree data is generated by scanning 3D shapes from 14 perspective and rotating the shape 12 times, they are rather more invariant with unaligned shapes than the data used in CAE-ELM. Specifically, we compute the relative improvement as $RI = Acc_{OCA-ELM} / Acc_{CAE-ELM} - 1$. The performance gain of OCA-ELM in aligned ModelNet10 is only 0.83%, while it improves the accuracy by 4.37% on unaligned ModelNet40. Hence, the contribution of the proposed algorithm is more significant in unaligned datasets.

TABLE 3.3: Comparison of CAE-ELM and OCA-ELM with different input

Input	ModelNet40	ModelNet10
CAE-ELM (voxel)	82.66%	88.66%
CAE-ELM (SDF)	80.11%	88.77%
CAE-ELM (voxel + SDF)	84.35%	91.41%
OCA-ELM (octree)	83.35%	89.57%
OCA-ELM (octree + voxel + SDF)	88.04%	92.17%

3.4 Summary

This chapter presents OCA-ELM, a combination of several discriminative representations and ELM for 3D shape classification. OCA-ELM takes advantage of features from different sources, particularly octree representations of 3D objects, which saves computational cost and enhances original ones with the normal vector of surfaces. ELM-AE and ELM classifiers are used for fast feature aggregation and fast learning. However, there are still some limitations of the proposed OCA-ELM. The aggregation of multiple inputs are simple concatenation, which ignores the prior knowledge (i.e., the relationship of features from the same input is stronger). OCA-ELM learns the whole training dataset in one during the training phase, the whole training dataset has to be loaded leading to tremendous memory cost. Therefore, once we want to train OCA-ELM with a newly coming training sample, the whole training dataset needs to be loaded again. Extensive experiments on benchmark datasets, namely ModelNet10 and ModelNet40, demonstrate the superior performance of OCA-ELM compared to previous methods.

Chapter 4

Unsupervised Feature Selection based Extreme Learning Machine for Clustering

Aiming to improve the clustering performance, Chapter 4 leverages unsupervised feature selection methods and proposes an extreme learning machine based clustering framework that directly derives clustering results without postprocessing. Section 4.2 describes the existing extreme learning machine based approaches for clustering. Section 4.3 explains the details of the proposed unsupervised feature selection based extreme learning machine for clustering. Section 4.4 evaluates the performance of the proposed method on the public benchmark datasets.

4.1 Background and Motivation

The original extreme learning machine (ELM) [55] can be seen as empirical risk minimization algorithm which tends to produce trivial solutions. To address this problem, regularized extreme learning machine [57] minimizes the sum error square of resident between target value and output along with the Frobenius norm of the output weights. Optimally pruned extreme learning machine(OP-ELM) [58] uses multiresponse sparse regression(MRSR) [59] to rank the hidden neurons, which can be considered as L1 regularization. TROP-ELM [60] extends OP-ELM [58] with Tikhonov-regularized PRESS to select the best hidden neurons, which is a solution with both L1 and L2 regularization. Nowadays, the machine learning tasks often come with a large-size feature set. For a specific task, some of the features are more relevant to the task while the rest are useless or even hurtful to the task. The data preprocessing that pick up those relevant features is called feature selection. Efficient robust feature selection(ERFS) [23] introduces a supervised feature selection method which jointly minimizes the L2,1 norm on both loss function and regularization. Such formulation results in a weight matrix are robust to the outliers of the data samples, where many rows shrink to zeros to achieve feature selection. Unsupervised discriminative feature selection(UDFS) [29] exploits scatter matrices and feature correlations simultaneously. The learned weight matrix is constrained by L2,1 norm and the features are selected according to the corresponding weight value. Similar to UDFS, Nonnegative discriminative feature selection(NDFS) [30] also adopt L2,1 norm regularization. The key difference between NDFS and UDFS is that NDFS proposes to learn a cluster indicator matrix with nonnegative constraint. Robust unsupervised feature selection (RUFS) [31] employs orthogonal-nonnegative matrix factorization to learn the pseudo labels and cluster centroids, which also selects features via L2,1 norms minimization. There are several existing ELM methods that adopt L2,1 norm regularization for supervised learning tasks[33–36], most of which minimizes the L2,1 norm of the output weight of ELM to select the best hidden neurons. Recently, ADELM [37] replaces random weight generation with autoencoder to improve the fingerprint classification performance, which also employs L2,1 norm to regularize the output weights.

In light of all these methods, we propose unsupervised feature selection based extreme learning machine (UFS-ELM) for clustering. We equip ELM with both L2,1 norm and manifold regularization to select hidden features and avoid overfitting.

Unlike the existing ELM algorithms, the proposed UFS-ELM is developed as an end-to-end solution for clustering and needs no post-processing to produce final clustering result. We then develop an iterative solution for the problem formulated. To be more specific, the L2,1 norm of output weights is embedded in the loss function and the intrinsic geometry of the output of ELM is constrained by manifold regularization. To the best of our knowledge, this is the first ELM method based on L2,1 norm regularization for clustering tasks.

4.2 Unsupervised Feature Selection based Extreme Learning Machine for Clustering

Generic data have different complicated distributions in their original spaces. The similar properties of data samples cannot be revealed easily. Therefore, it is often incomplete to cluster the generic data solely depending on their original features. Following the existing method [64, 134], we first use nonlinear mapping to transform the data into a hidden space, in which the similar properties of samples can be represented by the features. Then, the selected features can approximate the cluster indicator in the output space. Further, manifold regularization is exploited to retain the intrinsic geometry of the output space as similar to the one in the original space. Finally, the cluster indicator is computed from the output with both nonnegative and orthogonal constraints.

4.2.1 Objective Function

The optimization problem of UFS-ELM is formulated as below:

$$\begin{aligned} \min_{\beta, F} \quad & ||\bar{H}\beta - F||_F^2 + \lambda_1 ||\beta||_{2,1} + \lambda_2 \text{Tr}[\beta^T \bar{H}^T L \bar{H} \beta] \\ \text{s.t.} \quad & F \in [0, 1]^{n \times c} \end{aligned} \tag{4.1}$$

where $\bar{H}$ and β are the normalized hidden neurons and the output weights of ELM, respectively. L is the predefined Laplacian matrix and F is the cluster indicator matrix.

First, the random neurons in the hidden layer extend the original data by non-linear mapping and produce a large feature set H , which is then normalized to have zero mean. As stated by the ELM theory, the parameters of the randomly initialized neurons need not be modified. The original data matrix X is only used for constructing the normalized $\bar{H}$ and the Laplacian matrix L , which are fixed throughout the training process. The key idea of UFS-ELM is learning the output weights between the hidden layer and the output layer, so that the output of ELM $\bar{H}\beta$ can approximate the relaxed version of the cluster indicator matrix F as the mean square error between them is minimized.

Second, in ELM [54], the norm of the output weights β is minimized to maximize the separating margins of two different classes in the ELM feature space. In addition, the large feature set produced by the hidden layer possibly consist of worthless features. The L2,1 norm [23] is adopted to select the features that achieve better performance. The L2,1 norm of a matrix is defined as the L1 norm of the L2 norms of each row of the matrix. It happens to many unimportant features that all the weights corresponding to an individual feature will shrink to zero concurrently when the L2,1 norm of the output weights matrix β is minimized. By leaving only a small number of nonzero rows of β , the most significant features are selected. Thus, to embed the feature selection into the problem formulation and increase the separating margins, the L2,1 norm of output weights matrix is added into the optimization function. Together, the first two terms form a sparse regression model. The sparsity is controlled by the trade-off parameter λ_1 . If the value of λ_1 is extremely large, all the elements of β will shrink to zeros.

Last, the label smoothness assumption states that if two input data samples are close to each other, their corresponding labels are proportionally near. Thereby, manifold regularization [135] is used here to retain the intrinsic geometry of the data distribution, which is minimized. The output weights matrix has to be adjusted in such a way that the output $\bar{H}\beta$ has similar intrinsic geometry as the input matrix X .

As the problem formulated, the objectives of the proposed UFS-ELM are to learn clustering by adjusting the output weights to select features. As such, the output of ELM can approximate the cluster indicator matrix with similar geometry structure of the original data.

4.2.2 Alternating Optimization

According to [23], the L2,1 norm of β can be expressed in the form of $\text{Tr}[\beta^T B \beta]$ where B is an diagonal matrix with $B_{ii} = \frac{1}{\sqrt{\beta_i^T \beta_i + \epsilon}}$. Then the optimization problem becomes:

$$\begin{aligned} \min_{\beta, F} \quad & ||\bar{H}\beta - F||_F^2 + \lambda_1 \text{Tr}[\beta^T B \beta] + \lambda_2 \text{Tr}[\beta^T \bar{H}^T L \bar{H} \beta] \\ \text{s.t.} \quad & F \in [0, 1]^{n \times c} \end{aligned} \quad (4.2)$$

To solve the formulated problem, first, the cluster indicator F is initially obtained from k-means clustering on the original data. With the F fixed, the optimal solution of β can be obtained by setting the derivative of (4.2) with respect to β equal to zero

$$2\bar{H}^T(\bar{H}\beta - F) + 2\lambda_1 B\beta + 2\lambda_2 \bar{H}^T L \bar{H} \beta = 0 \quad (4.3)$$

thus,

$$\beta = Q^{-1} \bar{H}^T F \quad (4.4)$$

where $Q = \bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}$ is a symmetric matrix. Noted that B is initialized as an identity matrix and updated in every iteration after the update of β .

The loss function Eq. (4.2) can be rewritten as shown in Eq.(4.5).

$$\begin{aligned} & \text{Tr} [(\beta^T \bar{H}^T - F^T)(\bar{H}\beta - F) + \lambda_1 \beta^T B \beta + \lambda_2 \beta^T \bar{H}^T L \bar{H} \beta] \\ = & \text{Tr} [\beta^T \bar{H}^T \bar{H} \beta - \beta^T \bar{H}^T F - F^T \bar{H} \beta + F^T F + \lambda_1 \beta^T B \beta \\ & + \lambda_2 \beta^T \bar{H}^T L \bar{H} \beta] \\ = & \text{Tr} [\beta^T (\bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}) \beta - \beta^T \bar{H}^T F - F^T \bar{H} \beta \\ & + F^T F] \end{aligned} \quad (4.5)$$

We then substitute Eq. (4.4) and $Q = \bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}$ into Eq. (4.5)

$$\begin{aligned} & \text{Tr} [F^T \bar{H} Q^{-1} (\bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}) Q^{-1} \bar{H}^T F \\ & - 2F^T \bar{H} Q^{-1} \bar{H}^T F + F^T F] \\ = & \text{Tr} [F^T (\bar{H} Q^{-1} Q Q^{-1} \bar{H}^T - 2\bar{H} Q^{-1} \bar{H}^T + I_c) F] \\ = & \text{Tr} [F^T (\bar{H} Q^{-1} \bar{H}^T - 2\bar{H} Q^{-1} \bar{H}^T + I_c) F] \\ = & \text{Tr} [F^T (I_c - \bar{H} Q^{-1} \bar{H}^T) F] \end{aligned} \quad (4.6)$$

Thus, the optimization problem with respect to F becomes

$$\begin{aligned} \min_F \quad & \text{Tr} [F^T (I_c - \bar{H}Q^{-1}\bar{H}^T)F] \\ \text{s.t.} \quad & F \in [0, 1]^{n \times c} \end{aligned} \quad (4.7)$$

which can be solved by eigendecomposition on the matrix $I_c - \bar{H}Q^{-1}\bar{H}^T$. The c eigenvectors corresponding to the c smallest eigenvalues are the columns of F , where c is the number of clusters.

$$(I_c - \bar{H}Q^{-1}\bar{H}^T)\mathbf{v} = \mu\mathbf{v} \quad (4.8)$$

Let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c$ be the normalized eigenvectors corresponding to the smallest c eigenvalues of Eq. (4.8). The solution of Eq. (4.7) is

$$F = [\mathbf{v}_1^T, \mathbf{v}_2^T, \dots, \mathbf{v}_c^T]^T. \quad (4.9)$$

To fulfill the hard constraint of $F \in [0, 1]^{n \times c}$, winner-take-all is applied to each row of F . Specifically, all the elements of F are set to 0 except the maximum element in each row is set to 1. The solution for Eq. (4.2) is summarized in Algorithm 1.

4.3 Experimental Results and Discussions

In this section, two sets of experiments are conducted to evaluate the clustering capability of the proposed UFS-ELM algorithm and other ELM based clustering algorithms. First, the proposed UFS-ELM is compared with other ELM methods for clustering on the low-dimensional UCI datasets. The second set of experiments evaluate the methods on image and webpage high-dimensional datasets. In addition, the optimization problem of the UFS-ELM is composed by 3 terms with tuneable trade-off parameters, the sensitivity of the UFS-ELM to the trade-off parameters is also evaluated. Further, the approximation capability of ELM greatly depends on the number of hidden neurons, we evaluate effects of the number of hidden neurons in the UFS-ELM while fixing the other factors.

4.3.1 Datasets

Algorithm 1: The UFS-ELM algorithm.

Input: Normalized data $X \in \mathbb{R}^{n \times d}$,
Laplacian matrix $L \in \mathbb{R}^{n \times n}$,
number of clusters c ,
number of neurons m ,
trade-off parameters λ_1 and λ_2 ,
number of iterations t .

Output: Cluster indicator matrix $F \in \mathbb{R}^{n \times c}$.

begin
 Initialize the cluster indicator matrix F by K-means clustering on X .
 Generate input weights and biases of hidden neurons randomly and
 calculate the centered hidden layer output matrix $\bar{H}$.
 Initialize $B \in \mathbb{R}^{m \times m}$ as an identity matrix.
 Compute Q by $Q = \bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}$
 repeat
 Update $\beta = Q^{-1} \bar{H} F$
 foreach *diagonal element of B* **do**
 Update $B_{ii} = \frac{1}{\sqrt{\beta_i^T \beta_i + \epsilon}}$
 end
 Update $Q = \bar{H}^T \bar{H} + \lambda_1 B + \lambda_2 \bar{H}^T L \bar{H}$
 Update F by solving problem (4.7)
 until *Reaching t -th iteration.*
end

TABLE 4.1: Specifications of the Benchmark Datasets

Name	Samples	Features	Classes
Glass	214	9	6
Ionosphere	351	34	2
Pima	768	8	2
Vehicle	846	18	4
UMIST	575	644	20
warpPIE10P	210	2420	10
WebKBTexas	814	4029	7
YaleB	2414	1024	38
Robot	5456	24	4

The benchmark datasets are summarized in Table 4.1. UMIST [136], PIE10P [137] and YaleB [138] are face datasets, WebKBTexas [139] is a dataset contains web-pages collected from University of Texas. Robot Navigation dataset [140] is a large-scale ultrasound dataset. The rest are low-dimensional datasets from UCI machine learning repository.

UMIST face database has 564 images of 20 individuals. Each individual is shown in a range of poses from profile to frontal views. Each view is a 112×92 image and it is resized into 28×23 by using bicubic interpolation. The YaleB consists of 2414 frontal face images of 38 human subjects. Each image has 32×32 pixels. Due to the mixed lighting conditions, facial expressions, and occlusions, this dataset is much more challenging than the rest image datasets. WebKBTexas is a webpage dataset. The 814 pages are collected from the University of Texas and classified into 7 classes. 4029 words appeared in the webpages are stored in a dictionary. A binary value can be interpreted as absence/presence of a word in the dictionary. Each webpage sample is represented by a binary-value vector. warpPIE10P is also a face-image dataset of 10 persons.

Glass dataset consists of the information for classification of types of glass at the scene of crime. The study is motivated by criminological investigation. The features of Vehicle dataset is generated by the silhouettes of 4 vehicles, from different angles of view. Pima is the dataset contains medical predictor variables for the prediction of the outcome, such as BMI, insulin level, age, and so on. Ionosphere is the radar data collected by a antennas system, and the targets are free electrons in the ionosphere. Each data sample has 17 pulse numbers and each pulse number is described by 2 attributes. The Robot Navigation data is collected by 24 ultrasound sensors that surrounds the SCITOS G5 robot when the robot performed the wall-following task. 5456 sampling time steps are collected as data samples with a rate of 9 samples per second, and each sample corresponding to one of the four predetermined action categories (i.e. Move-Forward, Slight-Right-Turn, Sharp-Right-Turn, and Slight-Left-Turn). All the data are normalized by linearly rescaling each feature into the range of $[-1, 1]$.

4.3.2 Performance metric

The clustering accuracy is used as the performance metric to evaluate and compare the methods:

$$ACC = \frac{\sum_{i=1}^n \delta(y_i, \text{map}(y_i^*))}{n} \quad (4.10)$$

and

$$\delta(y_i, \text{map}(y_i^*)) = \begin{cases} 1 & y_i = y_i^* \\ 0 & \text{otherwise} \end{cases} \quad (4.11)$$

where $\text{map}(\cdot)$ is the best mapping function that maximizes the accuracy by Hungarian algorithm, which maps a predicted label to a true cluster label. y_i and y_i^* is the true cluster label and used to predict cluster label of the i -th sample; n is the total number of data samples.

The second performance metric is a normalized version of mutual information. For predicted label Y and ground truth Y^* , the metric can be computed as:

$$NMI' = \frac{H(Y) - H(Y|Y^*)}{\max\{H(Y), H(Y^*)\}} \quad (4.12)$$

where $H(Y)$ is the marginal entropy and $H(Y|Y^*)$ is the conditional entropy. Noted that this normalized version of mutual information is different from the well-known normalized mutual information shown in Eq.4.12:

$$NMI(Y, Y^*) = \frac{H(Y) - H(Y|Y^*)}{\sqrt{H(Y) * H(Y^*)}} \quad (4.13)$$

For the rest of this chapter, normalized mutual information refers to equation 4.13.

Purity is used as another measure of clustering performance, which can be defined as:

$$Purity(Y, Y^*) = \frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d| \quad (4.14)$$

where N is the number of data samples; M is the number of clusters in Y ; D is the number of classes in Y^* .

4.3.3 Experimental Setup

UFS-ELM is compared with several representative methods including ELM based methods for clustering:

- K-means, directly applies k-means on all the features of original data and the initialization of the cluster centroids are randomly picked;
- Unsupervised extreme learning machine (US-ELM) [65];
- ELMC^{LDA} [66], the nonlinear extension of LDA-KM based on the random feature mapping of ELM;
- Extreme learning machine for joint embedding and clustering (ELM-JEC) [64].

For all the ELM methods, the original setting of the related ELM methods [64, 65] is applied and set the number of hidden neurons to 2048 to facilitate the analysis of properties of UFS-ELM. Sigmoid is used as the nonlinear activation function of hidden neurons. The input weights and bias are initialized in the range $[-1, 1]$. When constructing the Laplacian matrix for manifold regularization, the size of neighbourhood is set at 5 or 10 for all datasets. The trade-off parameters for all methods are tuned by grid-search from $\{10^{-9}, 10^{-8}, \dots, 10^8, 10^9\}$. Each method is evaluated 10 times for every combination of above parameter setting. The best performance achieved by each method on each dataset is reported.

4.3.4 Comparison with Related Algorithms

The experimental results are summarized in Table 4.2, 4.3 and 4.4. Due to the different experimental setup, the results could be different from the results reported in the respective original papers. The best results for each dataset are in bold.

Not surprising that k-means clustering is outdone by all the ELM methods on the benchmark datasets, as mentioned, because the data in the original space can be nonlinear separable and cannot be grouped according to their Euclidean distance directly.

On the low-dimensional UCI datasets, US-ELM can achieve relatively good clustering accuracy and purity. However, it gives the largest standard deviation, which

means it is much more unstable on the low-dimensional datasets. ELM-JEC and ELMc^{LDA} obtain slightly worse results with low standard deviation, but they are still better than k-means. Thus, the embedding spaces generated by the ELM methods have better basis for k-means clustering, which is nonlinear transform of the original space. According to the results, UFS-ELM obtain dominant performance on the UCI datasets.

For the face-image datasets in Table 4.2, ELMc^{LDA} obtains the lowest clustering accuracy among ELM methods. ELMc^{LDA} converts the unsupervised learning clustering task into supervised learning problem by running k-means clustering on the hidden layer and using the results as pseudo label. The output weights is then trained on the pseudo label. Since the image data contains various noises due to the lighting conditions and occlusions, the pseudo label of ELMc^{LDA} is generated from the hidden layer mapped from the noisy data. Although the output weights are trained with regularization, the noise is not removed during the pseudo label generation. Therefore, ELMc^{LDA} is less effective to the face-image clustering.

The proposed method also imports cluster indicator matrix as pseudo label and trains the output weights based on it. It is able to achieve best performance on these face-image datasets. This is because the cluster indicator in the UFS-ELM is trained in each iteration, while the pseudo label in ELMc^{LDA} is computed only once and fixed. In this thesis, k-means clustering is used to initialize the cluster indicator and k-means clustering is not involved in the subsequent alternating solution. In another words, the cluster indicator and output weights are not trained to approximate the results of k-means clustering.

For WebKBTexas webpage datasets in Table 4.2, there is not much difference between the performances of ELM methods. The data format in WebKBTexas is presenting each webpage in a binary vector that indicates the presence of each word in the predefined dictionary. There is less noise with properly defined dictionary. However, the sequence and the frequency of the words are not expressed in this format. The data is simple and rather clear with less noise and redundancy. The performances are more different from each other in Table 4.3 and 4.4, ELM-JEC shows high clustering performance than other methods. UFS-ELM achieves second highest NMI with much lower variance.

YaleB face dataset has a large number of classes (i.e. individuals). As shown in Table 4.2, 4.3 and 4.4, UFS-ELM along with US-ELM consistently shows tier one performances. These results demonstrate the clustering ability of UFS-ELM on data with many classes. By conducting experiments on Robot Navigation, the objective is to verify the capability of the methods to handle large-scale dataset. The proposed UFS-ELM achieves the best clustering accuracy and purity, while the NMI is also comparable. Therefore, UFS-ELM shows good performance in handling large-scale data.

Overall, the proposed UFS-ELM presents superior performance on most of the datasets and competitive performance on the WebKBTexas dataset with rather low standard deviation. With respect to number of features and number of classes in different datasets, the performance of UFS-ELM has no clear trend as shown in Figure.4.1. The experimental results demonstrate the effectiveness of the proposed UFS-ELM.

FIGURE 4.1: Performance w.r.t. dataset attributes

TABLE 4.2: Clustering accuracy (%) on benchmark datasets

	K-means	US-ELM	ELMc ^{LDA}	ELM-JEC	UFS-ELM
Glass	43.81 $\pm$ 3.88	47.66 $\pm$ 7.49	45.65 $\pm$ 2.17	44.04 $\pm$ 1.47	50.00 $\pm$ 0.00
Ionosphere	71.08 $\pm$ 0.15	74.86 $\pm$ 7.66	71.74 $\pm$ 0.12	71.57 $\pm$ 0.18	80.77 $\pm$ 0.28
Pima	66.80 $\pm$ 0.03	67.38 $\pm$ 3.11	67.12 $\pm$ 0.07	66.36 $\pm$ 0.24	70.70 $\pm$ 0.96
Vehicle	37.26 $\pm$ 1.12	42.38 $\pm$ 5.86	41.02 $\pm$ 1.45	42.11 $\pm$ 0.24	51.98 $\pm$ 0.06
UMIST	43.83 $\pm$ 2.19	50.17 $\pm$ 2.42	46.04 $\pm$ 3.38	64.70 $\pm$ 2.41	76.74 $\pm$ 1.33
warpPIE10P	29.17 $\pm$ 2.04	45.60 $\pm$ 3.33	34.29 $\pm$ 2.40	48.45 $\pm$ 2.44	56.19 $\pm$ 0.00
WebKBTexas	57.94 $\pm$ 4.13	67.41 $\pm$ 2.03	67.94 $\pm$ 0.27	67.26 $\pm$ 0.24	67.75 $\pm$ 0.12
YaleB	9.88 $\pm$ 0.76	29.33 $\pm$ 2.54	15.03 $\pm$ 0.89	19.56 $\pm$ 0.60	31.59 $\pm$ 1.79
Robot	41.91 $\pm$ 1.72	43.21 $\pm$ 1.72	42.71 $\pm$ 1.72	44.68 $\pm$ 0.15	47.52 $\pm$ 0.58

TABLE 4.3: NMI (%) on benchmark datasets

	K-means	US-ELM	ELMc ^{LDA}	ELM-JEC	UFS-ELM
Glass	29.86 $\pm$ 3.46	17.69 $\pm$ 4.05	32.17 $\pm$ 2.24	22.37 $\pm$ 1.38	33.32 $\pm$ 0.00
Ionosphere	12.84 $\pm$ 0.25	14.51 $\pm$ 15.25	13.61 $\pm$ 0.21	13.38 $\pm$ 0.30	26.26 $\pm$ 0.97
Pima	5.12 $\pm$ 0.00	3.80 $\pm$ 4.58	5.29 $\pm$ 0.55	5.43 $\pm$ 0.54	11.53 $\pm$ 0.70
Vehicle	11.17 $\pm$ 2.44	14.33 $\pm$ 5.18	12.35 $\pm$ 2.34	12.39 $\pm$ 0.20	21.98 $\pm$ 0.36
UMIST	64.69 $\pm$ 1.25	66.77 $\pm$ 3.02	64.87 $\pm$ 2.76	82.04 $\pm$ 1.15	87.58 $\pm$ 1.50
warpPIE10P	28.81 $\pm$ 1.96	51.30 $\pm$ 5.51	34.78 $\pm$ 2.54	58.50 $\pm$ 1.86	64.91 $\pm$ 0.00
WebKBTexas	10.19 $\pm$ 5.15	12.04 $\pm$ 3.22	6.52 $\pm$ 1.69	15.81 $\pm$ 4.51	15.12 $\pm$ 0.47
YaleB	12.57 $\pm$ 1.18	36.62 $\pm$ 0.86	22.16 $\pm$ 1.53	22.42 $\pm$ 0.53	36.73 $\pm$ 0.94
Robot Navigation	10.88 $\pm$ 1.01	6.77 $\pm$ 2.18	11.61 $\pm$ 1.15	11.73 $\pm$ 0.33	11.27 $\pm$ 0.25

TABLE 4.4: Purity (%) on benchmark datasets

	K-means	US-ELM	ELMc ^{LDA}	ELM-JEC	UFS-ELM
Glass	53.50 ± 3.55	48.93 ± 3.74	55.51 ± 4.53	44.86 ± 0.00	63.55 ± 0.00
Ionosphere	71.05 ± 0.15	72.62 ± 7.82	71.74 ± 0.12	71.57 ± 0.18	80.48 ± 0.51
Pima	66.80 ± 0.00	68.05 ± 3.65	67.08 ± 0.12	66.97 ± 0.11	69.97 ± 0.76
Vehicle	40.33 ± 0.17	41.81 ± 5.27	39.80 ± 2.30	42.06 ± 0.44	51.48 ± 0.30
UMIST	50.42 ± 1.81	57.06 ± 3.80	51.25 ± 3.57	72.52 ± 2.47	81.04 ± 2.45
warpPIE10P	30.24 ± 2.46	50.10 ± 3.84	34.95 ± 1.97	52.33 ± 1.13	62.86 ± 0.00
WebKBTexas	69.08 ± 0.15	73.86 ± 3.65	69.36 ± 1.15	75.72 ± 2.91	69.30 ± 0.55
YaleB	10.25 ± 0.88	30.99 ± 1.22	16.42 ± 1.55	20.36 ± 0.64	31.78 ± 1.34
Robot Navigation	49.83 ± 1.21	47.16 ± 1.94	50.44 ± 1.25	50.56 ± 0.03	52.03 ± 1.31

4.3.5 Computational Complexity

This section investigates the computational complexity of the UFS-ELM and the related works with respect to the number of data samples n . The time complexity of K-means clustering is $O(n)$ [141]. The time complexity of US-ELM is $O(n^2)$ for finding the adjacency matrix. The rest adopted ELM methods involves eigendecomposition whose time complexity is $O(n^3)$. However, ELM-JEC and the UFS-ELM use eigendecomposition repeatedly to optimize the solution, while ELMc^{LDA} runs eigendecomposition only once during the supervised LDA. Moreover, the ELM-JEC stops its alternating optimization before the allowable number of iterations if the early stopping criterion was met. For the UFS-ELM, a maximum number of iterations is set as the stopping criterion. All experiments are repeated 10 times, the average running time are reported in Table 4.5:

TABLE 4.5: Running time on the benchmark datasets

	K-means	US-ELM	ELMc^{LDA}	ELM-JEC	UFS-ELM
Glass	0.01s	0.14s	0.15s	0.45s	0.77s
Ionosphere	0.01s	0.20s	0.12s	0.41s	0.94s
Pima	0.01s	0.15s	0.31s	0.56s	1.74s
Vehicle	0.01s	0.18s	0.66s	1.34s	2.21s
UMIST	0.03s	0.13s	0.36s	0.82s	1.68s
warpPIE10P	0.02s	0.09s	0.29s	2.62s	0.93s
WebKBTexas	0.54s	7.69s	38.28s	9.64s	8.02s
YaleB	0.26s	0.09s	2.47s	7.30s	7.78s
Robot Navigation	0.14s	1.91s	1.07s	15.26s	18.46s

As expected, k-means is the fastest method. US-ELM is slower than K-means but faster than the rest ELM methods, which is consistent with the complexity analysis. It is conspicuous that ELMc^{LDA} took 38.28s for WebKBTexas, as the ELMc^{LDA} applies a computationally expensive eigendecomposition on a matrix whose size is square of the number of features. The WebKBTexas has a large number of features. According to the analysis of complexity, ELM-JEC and UFS-ELM are more complex than US-ELM and ELMc^{LDA} . Empirically, UFS-ELM used more time than ELM-JEC on most datasets. As mentioned, there is no early stopping criterion in UFS-ELM. Instead, the number of iterations is predefined as a constant value of 10. Since both ELMJEC and UFS-ELM have complexity of $O(n^3)$, the running time is not the main concern. However, similar to ELMJEC, a early

stopping criterion depending on the clustering result or the cost function can be applied to shorten the actual running time of UFS-ELM.

4.3.6 Influence of Trade-off Parameters

There are two trade-off parameters in the UFS-ELM, which are the coefficients of manifold regularization and L2,1 norm of output weights in the formulation. As mentioned in the experimental setup, the performance of the UFS-ELM is evaluated with the trade-off parameters picked from $\{10^{-9}, 10^{-8}, \dots, 10^8, 10^9\}$. The clustering accuracy achieved by the optimal pairs of trade-off parameters values along with the neighbourhood are recorded, which are shown in Fig 4.2.

In Fig 4.2, on Glass and Pima datasets, the impressive performance can be observed over a large region around a range of 4 magnitudes for each parameter. On Vehicle and Ionosphere, significant performance can be achieved in a small region of the neighbourhood of the optima. The performance is changed more smoothly with the change of λ_2 . Thus, for clustering in low-dimensional data space, the trade-off parameters in the UFS-ELM should be carefully chosen especially λ_1 .

In Fig 4.2, the area of favourable value pairs is large on all datasets except UMIST. On UMIST, the adjacent area of optima is very rugged. The significant performances can be achieved in several nonadjacent points. Therefore, the performance of the UFS-ELM is less sensitive to the trade-off parameters on high-dimensional datasets.

FIGURE 4.2: Influence of trade-off parameters in UFS-ELM

4.3.7 Influence of Number of Hidden Neurons

This section continues to study the influence of number of hidden neurons in UFS-ELM. This can provide guidelines for using UFS-ELM. By fixing the rest factors at optima, this section evaluates the performance of UFS-ELM with different number of hidden neurons. Specifically, the number of dimensions of hidden layer is picked from $\{2^9, 2^{10}, 2^{11}, 2^{12}\}$. The results are shown in Fig 4.3.

For warpPIE10P dataset, the clustering accuracy remains the same when the number of hidden neurons changes in the predefined range. For the rest datasets, the accuracy performances kept increasing as the number of neurons is increased from 512 to 2048. It becomes stable when the number are more than 2048. Hence, the number of hidden neurons does not limit the capability of UFS-ELM once it is around 2048.

FIGURE 4.3: Influence of number of hidden neurons in UFS-ELM

4.3.8 Influence of L2,1 Norm

To verify the advance of L2,1 norm, we evaluate the accuracy performance of UFS-ELM with $\lambda_1 = 0$. The optimization problem becomes:

$$\begin{aligned} \min_{\beta, F} \quad & ||\bar{H}\beta - F||_F^2 + \lambda_2 \text{Tr}[\beta^T \bar{H}^T L \bar{H} \beta] \\ \text{s.t.} \quad & F \in [0, 1]^{n \times c} \end{aligned} \quad (4.15)$$

With $\lambda_1 = 0$, the inverse of Q could become not a number. To solve this, we add a small number (e.g. 10^{-30}) ϵ to the diagonal of Q as:

$$Q = \bar{H}^T \bar{H} + \epsilon I + \lambda_2 \bar{H}^T L \bar{H}$$

The variant is evaluated with the same experimental setup, and the best results are shown in Table 4.6. When $\lambda_1 = 0$, the accuracy performance remains the same on Glass and warpPIE10P, and decreases on the rest datasets. On the other hand, the variant has lower standard deviation on most of the datasets. These observations demonstrate that applying L2,1 norm in UFS-ELM could lead to better accuracy performance and a slightly loss in stability.

TABLE 4.6: Clustering accuracy of UFS-ELM with and without L2,1 norm

	$\lambda_1 > 0$	$\lambda_1 = 0$
Glass	50.00 $\pm$ 0.00	50.00 $\pm$ 0.00
Ionosphere	80.77 $\pm$ 0.28	78.06 $\pm$ 0.00
Pima	70.70 $\pm$ 0.96	64.45 $\pm$ 0.00
Vehicle	51.98 $\pm$ 0.06	51.89 $\pm$ 0.00
UMIST	76.74 $\pm$ 1.33	74.17 $\pm$ 2.65
warpPIE10P	56.19 $\pm$ 0.00	56.19 $\pm$ 0.00
WebKBTexas	67.75 $\pm$ 0.12	63.27 $\pm$ 0.00
YaleB	31.59 $\pm$ 1.79	30.77 $\pm$ 1.45
Robot Navigation	47.52 $\pm$ 0.58	43.86 $\pm$ 5.70

This thesis further replace the L2,1 norm regularizaiton in UFS-ELM with the L2 norm. By replacing $\mathbf{B}$ in Eq.4.2, the solution can be obtained in a similar way. The optimization problem becomes:

$$\begin{aligned} \min_{\beta, F} \quad & ||\bar{H}\beta - F||_F^2 + \lambda_1 \text{Tr}[\beta^T I \beta] + \lambda_2 \text{Tr}[\beta^T \bar{H}^T L \bar{H} \beta] \\ \text{s.t.} \quad & F \in [0, 1]^{n \times c} \end{aligned} \quad (4.16)$$

The best results achieved by the variant along with the results of UFS-ELM are shown in Table 4.7. The performances on Glass and warpPIE10P are not effected by changing the regularization. On UMIST dataset, the variant has higher accuracy than UFS-ELM while the standard deviation is also larger. UFS-ELM is slightly better than the variant on the rest datasets, the improvement of using L2,1 norm regularization is clear.

TABLE 4.7: Clustering accuracy of UFS-ELM with L2,1 norm and with L2 norm

	<i>L2,1norm</i>	<i>L2norm</i>
Glass	50.00 $\pm$ 0.00	50.00 $\pm$ 0.00
Ionosphere	80.77 $\pm$ 0.28	78.75 $\pm$ 0.38
Pima	70.70 $\pm$ 0.96	65.20 $\pm$ 0.20
Vehicle	51.98 $\pm$ 0.06	51.89 $\pm$ 0.00
UMIST	76.74 $\pm$ 1.33	76.89 $\pm$ 3.32
warpPIE10P	56.19 $\pm$ 0.00	56.19 $\pm$ 0.00
WebKBTexas	67.75 $\pm$ 0.12	67.65 $\pm$ 0.06
YaleB	31.59 $\pm$ 1.79	31.53 $\pm$ 0.60
Robot Navigation	47.52 $\pm$ 0.58	45.03 $\pm$ 3.45

4.4 Summary

This thesis proposes the unsupervised feature selection based extreme learning machine for clustering. Different from existing ELM methods for clustering, the proposed method directly produces the clustering results without using k-means clustering on the output of ELM. In addition, feature selection is embedded into the optimizing process by applying L2,1 norm to the output weights. The L2,1 norm forces the output weights to be sparse in row, thus removing the worthless features extracted by the hidden layer. Several experiments have been conducted to demonstrate the effectiveness of the proposed method. The computational complexity of the algorithms is also analyzed. At last, this thesis analyzes the influences of the trade-off parameters, the number of hidden neurons, and the L2,1 norm. Although unsupervised feature selection is applied, identification of the important features in the original data is not covered by the proposed method.

Chapter 5

Distance Fusion for Multiview Clustering

This chapter explores an innovative way to solve the key problem in multiview clustering - information fusion. Section 5.1 gives the background and motivation of this work. Section 5.2 describes the distance fusion and proposes a multiview clustering algorithm based on the distance fusion. Section 5.3 empirically evaluates the proposed methods and existing multiview clustering methods on a wide range of benchmark image datasets. This section also presents the convergence analysis of the proposed multiview clustering algorithm and the visualization of the distance measure.

5.1 Background and Motivation

Existing multiview spectral clustering methods fuse the information of graphs and eigenvectors to derive a consistent clustering result among multiple views. Recently, graph structure fusion (GSF) [105] learns a global graph of the multiview data. Specifically, GSF adopts the Hadamard product of the affinity matrices as the global affinity matrix. However, Hadamard product could remove the worthwhile edges while removing the redundant edges in the graphs. For example, two data samples are connected in most of the graphs but a single one. Regardless of how strong they are connected, the data pair will be disconnected during graph fusion.

Under this condition, the initial KNN graphs in GSF are usually constructed with a large k to reduce the number of disconnected pairs. This increases the redundant connections and reduces the locality of the data. Inspired by this, this chapter exploits a new way to fuse the structure information from multiple views in distance level namely distance fusion. Specifically, this thesis uses the mean of the distance matrices as the global distance matrix. To ensure a unified scale of the distances, the distance matrices are normalized before the between-view distance fusion. Row normalization and column normalization result in two different matrices and provide two perspectives of each pairwise distance. This thesis proposes within-view distance fusion to fuse these two matrices to include the metrics of both perspectives. In this way, each pairwise distance is compared with all the other pairwise distances related to the data pair. Unlike multi-kernel methods that only consider the data pair, the proposed similarity measure is obtained by taking into account the data pair and their neighbors. Our contributions are summarized as follows

- We propose within-view distance fusion to compute and combine the relative magnitudes of a pairwise distance with respect to the two paired data samples. By normalizing the pairwise distances with related distances, a more comprehensive measure of similarity can be obtained. Moreover, a unified scale of the distance matrices is ensured to facilitate the information fusion between multiple views.
- Under an unsupervised learning scenario, we assume different views are equally important. Besides, with the unified scale of distances provided by within-view distance fusion, there is no more worry about different scales. Therefore, we use between-view distance fusion to fuse multiview information in

a simple but effective way without bias toward any view. Based on this, we propose distance fusion with adaptive neighbors (DFAN) for multiview clustering. Unlike existing methods, DFAN fuses the multiview information in distance perspective before the construction of the similarity matrix. The global distance matrix is not adjusted during the learning of similarities.

- We conduct massive experiments to evaluate the proposed method and various state-of-the-art algorithms on several multiview datasets. We also empirically analyze the properties of the proposed method.

5.2 Proposed Methods

This section introduces the distance fusions and DFAN for multiview clustering.

5.2.1 Distance Fusion

For a multiview dataset $X_i = \{\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_n^T\}^T$, various metrics can be used to measure the distance between two samples. In this study, we adopt the squared Euclidean distance. For data samples $\mathbf{x}_i$ and $\mathbf{x}_j$, the squared Euclidean distance is defined as

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \quad (5.1)$$

Each data sample has n related distances, including the one linked to itself, and the local structure of the sample could be revealed from the rankings of its related distances (e.g., the K nearest neighbors). Yet, squared Euclidean distance is computed only based on the features of the paired samples without considering their respective local structures. In the local structures of two samples, the rankings of their pairwise distance could be different.

For example, consider points including x_1 , x_2 and x_3 as shown in Fig 5.1. We denote the distance between x_1 and x_2 as d_{12} , and denote the distance between x_2 and x_3 as d_{23} . In the perspective of x_1 , x_2 is further than some yellow points around x_1 . In the perspective of x_2 , x_1 is the closest point. Obviously, the ranking of this pairwise relationship for x_1 and x_2 are different. In other words, the

relative magnitudes of the distance among two sets of related distances are different. This relative magnitude provides us a more comprehensive lens to look at the pairwise relationship. It allows us to directly compare the distances between views. We obtain the relative magnitude of distance by dividing the square root of the sum of the squared distances, where the denominator serves as a reference for the magnitude of the distances. Normalization reveals the relative magnitude of this distance among the related distances. Combining two normalized versions of this distance includes the relative magnitude information and provides a more comprehensive measure of similarity. To achieve a single distance matrix for each view, the normalized distances are fused as:

$$\bar{d}_{ij} = \varphi\left(\frac{d_{ij}}{\|\mathbf{d}_i\|_2}, \frac{d_{ij}}{\|\mathbf{d}_j\|_2}\right) \quad (5.2)$$

where $\mathbf{d}_i$ is the i th row of the distance matrix; the within-view distance fusion function $\varphi(\cdot)$ can be arithmetic mean or geometry mean. Specifically, we can compute the distance matrix for each view as

$$\bar{d}_{ij} = \frac{1}{2}\left(\frac{d_{ij}}{\|\mathbf{d}_i\|_2} + \frac{d_{ij}}{\|\mathbf{d}_j\|_2}\right) = \frac{d_{ij} \cdot (\|\mathbf{d}_i\|_2 + \|\mathbf{d}_j\|_2)}{2 \cdot \|\mathbf{d}_i\|_2 \cdot \|\mathbf{d}_j\|_2} \quad (5.3)$$

or

$$\bar{d}_{ij} = \sqrt{\frac{d_{ij}}{\|\mathbf{d}_i\|_2} \cdot \frac{d_{ij}}{\|\mathbf{d}_j\|_2}} = \frac{d_{ij}}{\sqrt{\|\mathbf{d}_i\|_2 \cdot \|\mathbf{d}_j\|_2}} \quad (5.4)$$

Although a prominent goal of within-view distance fusion is to fuse the relative magnitudes of distances, it has another purpose. The ultimate goal of distance fusion is between-view information fusion. A lot of existing multiview clustering algorithms fuse the information between views while adjusting the weights of views. However, adjusting the weights of views during optimization would not make the global graph closer to the proper structure. Because, under an unsupervised learning scenario, we have no prior knowledge about which view is more important than the others. Adjusting the weights of views during optimization might leads to distortion of the final pairwise similarity. Even if a pairwise similarity is mismeasured in one view, this can be redressed by the rest views. So we assume the multiple views are equally important. Yet, the scales of multiview distance matrices could be different. Therefore, another purpose of within-view distance fusion is to ensure a unified scale of multiview distances. Hence, it is possible to combine distance matrices with equal weights. Consequently, the between-view distance fusion is

achieved by computing the arithmetic mean or geometric mean of the distance matrices:

$$\hat{d}_{ij} = \frac{1}{w} \sum_{v=1}^w \bar{d}_{ij(v)} \quad (5.5)$$

or

$$\hat{d}_{ij} = \left(\prod_{v=1}^w \bar{d}_{ij(v)} \right)^{\frac{1}{w}} \quad (5.6)$$

where $\bar{d}_{ij(v)}$ is the normalized pairwise distance from v th view, w is the total number of views. It is worth noting that the second form (i.e., Eq.(5.6)) of between-view fusion is a nonlinear combination of multiview distance matrices. We conduct experiments to compare this nonlinear fusion with conventional linear fusion.

FIGURE 5.1: Illustration of normalized distance. When measured by the Euclidean distances, x_1 and x_3 are equally far from x_2 (i.e., $d_{12} = d_{23}$). x_2 is the closest neighbor for x_3 and is further than most neighbors for x_2 , this difference is not expressed by Euclidean distance. When measuring the neighbors of x_2 by normalized distance, x_3 is closer than x_1 (i.e., $\bar{d}_{12} > \bar{d}_{23}$).

5.2.2 Distance Fusion with Adaptive neighbors

In this paper, we propose a multiview clustering methods

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} \quad & tr(\hat{D}^T A) + \alpha \|A\|_F^2 \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1, \\ & rank(L_A) = n - c \end{aligned} \quad (5.7)$$

where $\hat{D}$ is the fused distance matrix that computed by Eq.(5.5) or Eq.(5.6). α is the regularization parameter that controls the number of neighbors connected to each sample in A , which can be determined with adaptive neighbors. The formulation aims to learn an affinity matrix A with constrained Laplacian rank. We name this method as distance fusion with adaptive neighbors (DFAN) for multiview clustering. The proposed method is summarized in Algorithm 2.

Algorithm 2: Distance Fusion with Adaptive Neighbors

Require: Observed multiview data $\mathcal{X} = \{X_1, X_2, \dots, X_v\}$,number of clusters c ,number of nearest neighbors k .**Ensure:** Affinity matrix $A \in \mathbb{R}^{n \times n}$ with exact c connected components**for** $i = 1$ **to** v **do** Compute the pairwise distance D_i via Eq.(5.1); Compute the normalized distance $\bar{D}_i$ via Eq.(5.2);**end for**Compute the fused distance $\hat{D}$ via Eq.(5.5) or Eq.(5.6);Construct affinity matrix A from $\hat{D}$ Determine regularization parameter α via Eq.(5.22)**repeat** Construct Laplacian matrix L_A from A Compute class indicator matrix F from L via Eq.(5.9) Compute D_F from F via $d_{ij}^f = \|f_i - f_j\|_2^2$ Update A via Eq.(5.14)**until** the stopping criterion

5.3 Optimization

To solve the formulated optimization problem, we begin with the Laplacian rank constraint. The Laplacian rank constraint in Eq.(5.7) can be secured by making the smallest c eigenvalues equal to zero. Since L_A is positive semidefinite, all the eigenvalues are nonnegative. Therefore, the constraint becomes

$$\sum_{i=1}^c \sigma_i(L_A) = 0$$

where $\sigma_i(L_A)$ is the i th smallest eigenvalue of (L_A) . According to Ky Fan's Theorem [142], we have

$$\sum_{i=1}^c \sigma_i(L_A) = \min_{F \in \mathbb{R}^{n \times c}, F^T F = I} \text{tr}(F^T L_A F)$$

Thus, the optimization problem Eq.(5.7) is equivalent to

$$\begin{aligned} \min_{A, F} \quad & \text{tr}(\hat{D}^T A) + \alpha \|A\|_F^2 + \lambda \text{tr}(F^T L_A F) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \tag{5.8}$$

With alternating optimization [143], the problem can be solved by updating A and F iteratively.

5.3.1 Update F with fixed A

To update the cluster indicator matrix F , we fix the global affinity matrix A . Then, the problem becomes a single-view clustering problem and Eq.(5.8) turns into the optimization problem of spectral clustering Eq.(5.9). In spectral clustering, F is computed as the c eigenvectors corresponding to the largest c eigenvalues:

$$\min_{F \in \mathbb{R}^{n \times c}, F^T F = I} \text{tr}(F^T L F) \quad (5.9)$$

5.3.2 Update A with fixed F

When F is fixed, Eq.(5.8) becomes

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} \quad & \text{tr}(\hat{D}^T A + \alpha A^T A + 2\lambda F^T L_A F) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (5.10)$$

With

$$\sum_{i,j}^n \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 a_{ij} = 2\text{tr}(F^T L_A F) \quad (5.11)$$

and λ is the trade-off parameter auto-adjusted during the optimization. Thus, Eq.(5.10) can be rewritten as

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} \quad & \sum_{i,j} (\hat{d}_{ij} a_{ij} + \alpha a_{ij}^2 + \lambda d_{ij}^f a_{ij}) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (5.12)$$

where $d_{ij}^f = \|\mathbf{f}_i - \mathbf{f}_j\|_2^2$. Since Eq.(5.12) is independent between different rows of A , the problem can be further simplified as

$$\begin{aligned} \min_{\mathbf{a}_i} \quad & \sum_{j=1}^n (\hat{d}_{ij} a_{ij} + \alpha a_{ij}^2 + \lambda d_{ij}^f a_{ij}) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (5.13)$$

Let the derivative of Eq.(5.13) equal to zero, we have

$$a_{ij} = (\hat{d}_{ij} + \lambda d_{ij}^f)/2\alpha \quad (5.14)$$

5.3.3 Initialization

In the first place, the affinity matrix A is initialized solely based on the global distance matrix $\hat{D}$ without considering the rank of L_A . The initial A is computed as

$$a_i = \hat{d}_i/2\alpha \quad (5.15)$$

where the regularization parameter α is hard to tune since it can be assigned with any nonnegative value without explicit meaning. The purpose of this regularization is to control the number of neighbors connected to each sample. According to [20], the finest value for α can be determined based on the adaptive neighbors. The Lagrangian expression of Eq.(5.13) is as follows

$$\begin{aligned} \mathcal{L}(a_i, \eta, \beta_i) = & \sum_{j=1}^n (\hat{d}_{ij} a_{ij} + \alpha a_{ij}^2) \\ & - \eta(a_i^T \mathbf{1} - 1) - \beta_i^T \mathbf{a}_i \end{aligned} \quad (5.16)$$

where $\eta, \beta_i \geq 0$ are the Lagrangian multipliers. According to the KKT condition [144], a_{ij} can be optimized as

$$a_{ij} = \max(0, \eta - \frac{\hat{d}_{ij}}{2\alpha}) \quad (5.17)$$

Let $d_{i,k}$ indicate the k th smallest elements in $\mathbf{d}_i$. In order to obtain a sparse affinity matrix A with each row has only k connections corresponding to the adaptive neighbors, each row should has k nonnegative elements. Therefore, from Eq.(5.17), we have

$$\begin{cases} a_{i,k} > 0 \\ a_{i,k+1} = 0 \end{cases} \quad (5.18)$$

thus,

$$\begin{cases} \eta - d_{i,k}/2\alpha > 0 \\ \eta - d_{i,k+1}/2\alpha \leq 0 \end{cases} \quad (5.19)$$

Based on the constraint $\mathbf{a}_i^T \mathbf{1} = 1$, we have

$$\begin{aligned} 1 \sum_{j=1}^k (\eta - d_{i,k}/2\alpha) \\ \Rightarrow \eta = \frac{1}{k} + \frac{1}{2k\alpha} \sum_{j=1}^k \hat{d}_{ij} \end{aligned} \quad (5.20)$$

substitute Eq.(5.20) into Eq.(5.19), we have

$$\begin{cases} \frac{k}{2} \hat{d}_{i,k+1} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} \geq \alpha \\ \frac{k}{2} \hat{d}_{i,k} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} < \alpha \end{cases} \quad (5.21)$$

The optimal solution of Eq.(5.13) with exact k nonnegative elements can be obtained with the maximal solution of Eq.(5.21)

$$\alpha = \frac{k}{2} \hat{d}_{i,k+1} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} \quad (5.22)$$

It is obvious that a specific α can be calculated for each $\hat{\mathbf{d}}_i$. Therefore we set α as the arithmetic mean of all the candidates.

5.3.4 Parameters and Stopping Criterion

During the alternating optimization, while the affinity matrix A and the cluster indicator matrix F are optimized, the trade-off parameter λ is also updated. λ is updated in each iteration depending on the sum of the eigenvectors. λ is doubled if the sum of the c eigenvectors corresponding to the smallest c eigenvalues is less than a predefined threshold. λ is decreased if the sum of the $c + 1$ eigenvectors corresponding to the smallest $c + 1$ eigenvalues is greater than the predefined threshold. The stopping criterion of the algorithm also refers to the sum of the eigenvectors and λ . The algorithm stops when λ stops updating.

5.4 Experiments

We conduct experiments on several multiview datasets to evaluate the clustering performances of DFAN and various multiview clustering methods. The properties of DFAN are also analyzed.

5.4.1 Datasets

To evaluate the multiview clustering performance of DFAN and related methods, we adopt three multiview datasets in this study. The number of features in each view and other information of the datasets are shown in [5.1](#).

TABLE 5.1: Specifications of the Datasets

Dataset	Handwritten	MSRCv1	COIL20	BBC
Classes	10	9	20	2
Samples	2000	240	1440	145
View 1	76	1302	1024	4633
View 2	216	48	944	4659
View 3	64	512	324	4665
View 4	240	100	-	4684
View 5	47	256	-	-
View 6	6	-	-	-

FIGURE 5.2: Samples of Handwritten Digits

FIGURE 5.3: Samples of MSRCv1

FIGURE 5.4: Samples of COIL20

5.4.1.1 Handwritten Digits

An image dataset [145] consists of six features of handwritten digits (0 to 9) extracted from a collection of Dutch utility maps. 200 samples for each digit have been digitized in binary images, which are represented in terms of the following six kinds of the feature: 76 Fourier coefficients of the character shapes, 216 profile correlations, 64 Karhunen-Love coefficients, 240 tiles of 2x3 pixels and counting the number of object pixels in each tile as shown in Fig. 5.2(a), 47 rotation invariant Zernike moments [146], and six morphological features like the number of endpoints of the skeleton.

5.4.1.2 MSRCv1

A dataset contains 240 images in 9 classes as a whole. Following [72], we select seven classes: tree, building, airplane, cow, face, car, and bicycle. The first view is the 1302-D CENTRIST feature, the second is the 48-D color moment, the third is the 512-D GIST feature, the fourth is the 100-D HOG feature, the fifth view is the 256-D LBP feature, and the sixth view is 200-D SIFT feature. The sample images of the seven classes are shown in Fig. 5.3.

5.4.1.3 COIL20

The original image dataset COIL20 [147] is available in two versions (unprocessed and processed), the processed version is adopted in this study. The dataset includes 1440 images of 20 different objects. The images consist of the smallest square that contains the object, and the background is removed. Each image is resized to 32×32 pixels as the 1024-D grayscale feature. The other two features are 944-D local binary pattern (LBP) feature and 324-D histogram of oriented gradients (HOG) feature extracted from the original images. The sizes of LBP cell and HOG cell are 32×32 , and L2 normalization is applied to each LBP cell histogram. The samples of the processed version dataset are shown in Fig. 5.4.

5.4.1.4 BBC

A document dataset [148] consists of 145 document samples. The samples have 4 views by splitting each document into four related segments. Each segment contains at least 200 characters and is constituted by consecutive textual paragraphs.

5.4.2 Experimental Setup

Besides the proposed DFAN, we also evaluate several competitive multiview clustering methods, as shown below:

CMSC: Co-regularized multi-view spectral clustering [103] computes a set of eigenvectors for each view as the cluster indicator matrix by single-view spectral clustering and regularizes each view-specific cluster indicator toward a common centroid cluster indicator. CMSC does not have many hyperparameter to be tuned and save the tuning time for different datasets. Yet, the eigenvectors obtained from different view have different scale making them incomparable. If scales of different views various, the clustering result might not be reliable.

MMSC: Multi-modal spectral clustering [149] treats each feature set of multiview data as a modal, and a commonly shared graph Laplacian matrix is learned by uniting multiple modals. The single-view spectral clustering error is minimized to learn a cluster indicator matrix for each view. The single multi-modal cluster indicator is paired with each view. The sum of all pairwise matrix distances is minimized to adjust the multi-modal cluster indicator. MMSC fuses the highly abstracted graphs, the fusion is quite straight forward. However, fusion in the late stage wastes a lot of information.

MLAN: Multi-view learning with adaptive neighbors [104] learns the local structure of multiview data and the cluster indicator simultaneously. During the learning of the global graph, the weight for each view is adjusted automatically. However, the weights for views are not adjusted based on the importances of views, the weights are optimized to minimize the difference between views. With constrained Laplacian rank, the optimal graph has the number of components equals the predefined number of clusters.

MCGC: Similar to MMSC, multiview consensus graph clustering [72] learns the view-specific cluster indicators and a global graph simultaneously. During the optimization, the cluster indicators are weighted by predefined trade-off parameters. The multiview fusion is performed on the cluster indicators in the late stage.

PMSC: Partition level multiview subspace clustering [150] represents each data points as a linear combination of other points to learn an affinity graph for each view. The partition results are produced for each view by spectral clustering. PMSC defines the distance between partition results and minimizes the distances between the global partition result and other partition results to acquire the optimal global partition.

GSF: Graph structure fusion [105] adopts Hadamard product of the view-specific affinity matrices as the global affinity matrix. A global graph is adaptively learned from the single affinity matrix by spectral clustering with Laplacian rank constraint. The zero connections are emphasized by the Hadamard product.

SwMC: With precomputed view-specific similarity matrices as the input, self-weighted multiview clustering with multiple graphs (SwMC) [89] computed the difference between two matrices. SwMC then minimizes difference between global matrix and each view-specific similarity matrix. The weight for each matrix distance is defined as a function of the matrix distance itself, which is automatically adjusted during the learning of the global similarity matrix.

MATLAB `normalize()` function is used to obtain data with center 0 and standard deviation 1. For KNN graphs construction, if data samples of the same class are close in a particular space, a small K will achieve good performance. On the other hand, the computation load noticeable increases as the value of K increases. To make a fair comparison and reduce the computation load, the number of nearest neighbors is picked from $\{3, 4, 5, 6, 7, 8, 9, 10\}$. For the trade-off parameters, we follow the experimental setup in the original papers. The co-regularization parameter λ for CMSC is varied from 0.01 to 0.05. For MMSC, we fix the penalty parameter $\log_{10} \alpha$ with α in the range from -2 to 2 with incremental step 0.2. In the original paper, the parameter β of MCGC is set to 0.6. In this study, the value of β is varied in $\{0.01, 0.1, 0.6, 1, 10, 100\}$. The experiments of each setting is repeated ten times, and the best averages of the results are reported.

5.4.3 Evaluation Metrics

The clustering accuracy is used as the performance metric to evaluate and compare the methods:

$$ACC = \frac{\sum_{i=1}^n \delta(y_i, \text{map}(y_i^*))}{n} \quad (5.23)$$

and

$$\delta(y_i, \text{map}(y_i^*)) = \begin{cases} 1 & y_i = y_i^* \\ 0 & \text{otherwise} \end{cases} \quad (5.24)$$

where $\text{map}(\cdot)$ is the best mapping function that maximizes the accuracy by Hungarian algorithm [151], which maps a predicted label to a true cluster label. y_i and y_i^* is the true cluster label and predicted cluster label of the i th sample, and n is the total number of data samples.

The second performance metric is a normalized version of mutual information. For predicted label Y and ground truth Y^* , the metric can be computed as:

$$NMI(Y, Y^*) = \frac{H(Y) - H(Y|Y^*)}{\sqrt{H(Y) * H(Y^*)}} \quad (5.25)$$

Purity is used as another measure of clustering performance, which can be defined as:

$$Purity(Y, Y^*) = \frac{1}{n} \sum_{m \in c} \max_{d \in c^*} |m \cap d| \quad (5.26)$$

where n is the number of data samples, c is the number of clusters in Y , c^* is the number of classes in Y^* .

An alternative interpretation of clustering is to view it as $n(n-1)/2$ pairwise decisions. For a pair of samples to have the same ground truth label, their pairwise decision is a true positive (TP) if they are grouped in the same cluster. Otherwise, the decision is a false negative (FN). For a pair of samples that have different ground-truth labels, their pairwise decision is false positive (FP) if they are grouped in the same cluster. Otherwise, the decision is true negative (TN). Precision and recall are defined as

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5.27)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5.28)$$

and the default balanced F-score is

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.29)$$

Rand index is the percentage of correct decisions among all decisions:

$$\text{Rand index} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (5.30)$$

The adjusted Rand index (ARI) is the corrected-for-chance version of the Rand index, which is defined as

$$\text{ARI} = \frac{\sum_{ij} C_{n_{ij}}^2 - (\sum_i C_{a_i}^2 \sum_j C_{b_j}^2) / C_n^2}{(\sum_i C_{a_i}^2 + \sum_j C_{b_j}^2) / 2 - (\sum_i C_{a_i}^2 \sum_j C_{b_j}^2) / C_n^2} \quad (5.31)$$

5.4.4 Clustering Results

The experimental results of proposed method and the related multiview clustering methods on the three datasets are listed in Table 5.2, Table 5.3 and Table 5.4 separately.

CMSC has the worst overall performance on all the three datasets. The most crucial difference between CMSC and the rest methods is the timing of fusion. CMSC only fuses the multiple views after obtaining the view-specific cluster indicators via spectral clustering, while the rest methods fuse the multiview information before the eigendecomposition of spectral clustering. Therefore, it is clear that post-clustering fusion produces worse results. This is due to the loss of information during the clustering. Besides, the results acquired by this method come with the highest variation.

In most comparisons, MLAN has better performances than MMSC and MCGC. This is expected since they all use the weighted sum of multiple graphs as the global graph, while MLAN adjusts the weights during the optimization and the rest use fixed weights.

In Table 5.2, the proposed DFAN yielded the best performances in every metric except precision. DFAN has a precision that slightly lower than MCGC but still better than the rest methods. In Table 5.3, the proposed DFAN also loses to MCGC in precision and has a similar F-score as the MLAN. For the rest performance

metrics, DFAN obtains the best scores. In Table 5.4, DFAN secures the highest values in all metrics. DFAN utilizes the equal weights for the views and similar spectral clustering to find the clusters. The differences between DFAN and the above three methods are, DFAN employs nonlinear combination and two-stage distance fusion. The performances of DFAN have been proved the effectiveness of between-view distance fusion and within-view distance fusion with normalization.

TABLE 5.2: Clustering Performance (%) on Handwritten Digits

	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
CMSC	87.00 ± 7.38	82.06 ± 4.73	87.50 ± 6.61	79.01 ± 4.54	80.98 ± 7.11	79.96 ± 7.97	77.70 ± 7.95
MMSC	93.98 ± 5.77	93.60 ± 2.28	94.49 ± 4.95	90.28 ± 2.70	94.16 ± 5.24	92.00 ± 7.94	91.07 ± 5.89
MLAN	97.40 ± 0.00	94.29 ± 0.00	97.40 ± 0.00	94.78 ± 0.27	95.21 ± 0.00	94.85 ± 0.00	94.28 ± 0.00
MCGC	95.85 ± 0.00	91.36 ± 0.00	95.85 ± 0.00	91.71 ± 0.00	99.12 ± 0.00	91.95 ± 0.00	91.05 ± 0.00
PMSC	67.62 ± 6.82	72.95 ± 3.17	71.17 ± 6.40	53.43 ± 0.00	99.11 ± 6.28	60.73 ± 4.52	55.73 ± 6.97
GSF	96.50 ± 0.00	92.68 ± 0.00	96.50 ± 0.00	92.98 ± 0.00	93.24 ± 0.00	93.11 ± 0.00	92.35 ± 0.00
SwMC	84.27 ± 4.59	87.13 ± 2.41	86.78 ± 3.46	76.60 ± 6.34	90.87 ± 4.71	81.46 ± 6.84	79.25 ± 5.32
DFAN	98.10 ± 0.00	95.59 ± 0.00	98.10 ± 0.00	96.13 ± 0.00	98.02 ± 0.00	96.19 ± 0.00	95.77 ± 0.00

TABLE 5.3: Clustering Performance (%) on MSRCv1

	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
CMSC	68.90 ± 10.80	62.96 ± 6.21	72.19 ± 8.63	57.62 ± 8.50	61.18 ± 8.83	59.33 ± 9.14	52.52 ± 10.42
MMSC	77.62 ± 4.67	72.79 ± 1.75	80.62 ± 3.17	66.19 ± 1.17	72.51 ± 3.83	68.94 ± 4.83	63.67 ± 4.55
MLAN	81.90 ± 0.00	77.02 ± 0.00	81.90 ± 0.00	70.10 ± 0.00	77.60 ± 0.00	73.66 ± 0.00	69.16 ± 0.00
MCGC	74.29 ± 0.00	67.22 ± 0.00	76.19 ± 0.00	61.60 ± 0.00	94.38 ± 0.00	64.02 ± 0.00	57.82 ± 0.00
PMSC	70.71 ± 0.34	65.44 ± 2.70	73.10 ± 3.70	53.89 ± 7.04	66.19 ± 6.02	58.86 ± 5.54	51.52 ± 7.01
GSF	79.52 ± 0.00	73.42 ± 0.00	80.95 ± 0.00	67.92 ± 0.00	76.06 ± 0.00	70.44 ± 0.00	65.26 ± 0.00
SwMC	75.33 ± 3.61	73.03 ± 5.70	78.14 ± 2.40	61.14 ± 7.10	74.92 ± 6.26	66.41 ± 4.05	60.29 ± 7.47
DFAN	82.86 ± 0.00	78.05 ± 0.00	82.86 ± 0.00	69.67 ± 0.00	82.40 ± 0.00	75.25 ± 0.00	70.86 ± 0.00

TABLE 5.4: Clustering Performance (%) on COIL20

	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
CMSC	71.58 ± 4.41	83.17 ± 1.27	74.04 ± 3.11	66.77 ± 2.35	72.24 ± 2.82	69.35 ± 3.92	67.70 ± 3.00
MMSC	76.03 ± 6.34	91.65 ± 2.07	80.20 ± 4.98	67.69 ± 3.73	89.53 ± 5.02	75.27 ± 3.64	73.76 ± 5.37
MLAN	87.50 ± 0.00	96.64 ± 0.00	89.86 ± 0.00	82.29 ± 0.00	96.83 ± 0.00	88.97 ± 0.00	88.35 ± 0.00
MCGC	73.82 ± 0.00	86.89 ± 0.00	78.06 ± 0.00	63.13 ± 0.00	81.10 ± 0.00	70.15 ± 0.00	68.40 ± 0.00
PMSC	35.17 ± 0.25	60.18 ± 2.93	43.40 ± 1.18	11.92 ± 1.06	93.72 ± 2.20	20.42 ± 1.51	13.06 ± 2.56
GSF	84.72 ± 0.00	94.54 ± 0.00	85.00 ± 0.00	81.00 ± 0.00	94.31 ± 0.00	85.11 ± 0.00	84.30 ± 0.00
SwMC	72.94 ± 3.54	85.49 ± 2.07	77.01 ± 3.46	59.59 ± 1.46	91.13 ± 3.38	67.53 ± 3.72	65.60 ± 3.61
DFAN	99.86 ± 0.00	99.79 ± 0.00	99.86 ± 0.00	99.72 ± 0.00	99.73 ± 0.00	99.72 ± 0.00	99.71 ± 0.00

TABLE 5.5: Clustering Performance (%) on BBC

	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
CMSC	92.93 ± 3.10	22.30 ± 15.49	93.87 ± 1.40	88.90 ± 1.57	97.82 ± 2.96	92.77 ± 3.49	26.85 ± 28.04
MMSC	93.10 ± 0.00	17.19 ± 0.00	93.10 ± 0.00	89.56 ± 0.00	99.89 ± 0.00	92.99 ± 0.00	24.14 ± 0.00
MLAN	89.66 ± 0.00	1.20 ± 0.00	92.41 ± 0.00	85.52 ± 0.00	94.20 ± 0.00	89.65 ± 0.00	-3.79 ± 0.00
MCGC	93.79 ± 0.00	26.33 ± 0.00	93.79 ± 0.00	88.12 ± 0.00	99.80 ± 0.00	93.60 ± 0.00	27.11 ± 0.00
PMSC	92.07 ± 0.49	5.12 ± 5.22	92.41 ± 0.00	88.23 ± 0.08	98.46 ± 0.41	92.00 ± 2.47	11.51 ± 8.04
GSF	88.97 ± 0.00	20.41 ± 0.00	92.41 ± 0.00	95.23 ± 0.00	92.81 ± 0.00	88.97 ± 0.00	16.24 ± 0.00
SwMC	92.69 ± 1.42	16.11 ± 13.19	93.24 ± 0.71	87.15 ± 1.42	98.70 ± 1.33	92.57 ± 1.25	15.39 ± 15.13
DFAN	93.79 ± 0.00	26.33 ± 0.00	93.79 ± 0.00	88.12 ± 0.00	99.80 ± 0.00	93.60 ± 0.00	27.11 ± 0.00

FIGURE 5.5: Distance matrices of HD

FIGURE 5.6: Distance matrices of MSRCv1

FIGURE 5.7: Distance matrices of COIL20

5.4.5 Convergence Analysis

The proposed method finds local optima of Eq. (5.8) by iteratively adjusting A and F . When we update F with fixed A , the problem becomes a single-view spectral clustering problem Eq. (5.9). Since L_A is positive semidefinite and Eq. (5.9) is a convex quadratic function, a local minimum can be found. When we update A with fixed F , the problem becomes Eq. 5.13, which is also a convex quadratic function. Besides, we empirically show the convergence of the proposed method by plotting the loss function against the iteration number. The convergence curves of the loss function value are shown in Fig. 5.8. We remove the stopping criterion to plot the curves until 30 iterations. In all cases, the loss function converged within 15 iterations.

(A) HD

(B) MSRCv1

(C) COIL20

(D) BBC

FIGURE 5.8: Convergence Curves of Loss Function

5.4.6 Different Forms of Distance Fusions

Both within-view distance fusion and between-view distance fusion have two forms: arithmetic mean and geometric mean. This section evaluates the proposed DFAN with combinations of different forms of distance fusions. The best performances of each form are reported in Table 5.6 and Table 5.7.

In Table 5.6, within-view distance fusion with geometric mean obtains comparable performance on MSRCv1 and higher scores on the other two datasets. Compared to the other methods listed in Table 5.2, Table 5.3 and Table 5.4, DFAN with arithmetic mean within-view distance fusion still gains dominant performance. Based on the experimental results, rather than arithmetic mean, geometric mean is the more promising solution for within-view distance fusion.

In Table 5.7, the performances of the two forms are close on MSRCv1, and each of them wins the race on one dataset. Thus, there is no clear evidence that one form is always better than the other. The multiview data may consist of some meaningless or even misleading feature sets. It is unknown that the magnitudes of the misleading features are large or small. Depending on the magnitudes of the misleading features, the two forms of between-view distance fusion may emphasize or despise them. On the other hand, for within-view distance fusion, both row-normalized and column-normalized distances are meaningful. Therefore, both forms of within-view distance fusion can achieve significant performances.

TABLE 5.6: Clustering Performance (%) of DFAN with different within-view distance fusions

Handwritten Digits	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	97.60	94.66	97.60	95.14	96.16	95.23	94.70
GM	98.10	95.59	98.10	96.13	96.26	96.19	95.77
MSRCv1	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	82.86	78.05	82.86	69.80	82.40	75.25	70.86
GM	82.86	76.77	82.86	69.80	81.38	75.15	70.78
COIL20	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	97.01	98.35	97.01	94.14	97.33	95.66	95.43
GM	99.86	99.79	99.86	99.72	99.73	99.72	99.71

AM: arithmetic mean; GM: geometric mean.

TABLE 5.7: Clustering Performance (%) of DFAN with different between-view distance fusions

Handwritten Digits	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	98.10	95.59	98.10	96.13	96.26	96.19	95.77
GM	77.10	84.69	77.15	66.87	96.16	76.62	73.61
MSRCv1	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	82.38	78.05	82.38	69.25	82.40	75.25	70.86
GM	82.86	76.77	82.86	69.80	81.38	75.15	70.78
COIL20	Accuracy	NMI	Purity	F-score	Precision	Recall	ARI
AM	92.78	98.00	94.79	90.21	98.44	93.63	93.29
GM	99.86	99.79	99.86	99.72	99.73	99.72	99.71

AM: arithmetic mean; GM: geometric mean.

5.4.7 Distance Matrix Analysis

This section displays each distance matrix as a squared image, and each pixel in the images represents a pairwise distance. The images use the full range of colors in a predefined colormap. The brighter color indicates a larger distance, and a smaller distance is marked in a darker color. The smallest values map to dark blue, and the greatest values map to the bright yellow. The ideal distance matrix enlarges the distances of points from different clusters and minimizes the distances of points within the same cluster. Specifically, after sorting the samples based on their ground-truth labels, there are dark blue blocks on the diagonal of the ideal distance matrices. The distance matrices produced from the datasets via distance fusion are shown in Fig. 5.5, Fig. 5.6 and Fig. 5.7. The squared images in the first rows are produced without within-view distance fusion. The second rows are produced with within-view distance fusion. The last columns are produced with between-view distance fusion. The rest columns are the view-specific distances.

The influence of within-view distance fusion can be revealed by comparing the rows. Overall, the second rows are brighter than the first rows. The blocks away from the diagonals are brighter in the second rows. This means within-view distance fusion enlarges the distances between different classes. Still, the blocks on the diagonals remain in the dark, which means the samples in the same class stick together after the fusions. Especially in Fig. 5.7a and Fig. 5.7b, almost all the pixels are in dark. After the fusions, the dark blocks on the diagonals become more distinguishable in Fig. 5.7e and Fig. 5.7f. In other words, the fusion further split different classes while keeping the distances within the same class small.

By comparing to the ideal distances, the misleading view-specific distance matrices can be easily found. In Fig. 5.6a, there is no clear dark blue block but only a narrow blue line on the diagonal. In this case, the samples within the same class are not similar, and the clusters might not be found based on this single view. In Fig. 5.5d, some dark blocks appear in the corner that far from the diagonal. Those blocks mean some pairs from different classes have small distances, leading to connections of these pairs in the graph. Another case is that there is no clear boundary in the figure (e.g., Fig. 5.5b). Such views are more or less useless. Therefore, some views provide misleading features, and the graphs or clustering results based on these view-specific distances could be wrong. On the other hand, global distances are produced by fusing all the views and more like the ideal distances.

5.5 Summary

This chapter proposes to fuse multiview information in the distance level. The within-view distance fusion computes and fuses two relative magnitudes of the distance to obtain a more comprehensive similarity measure. The resulting view-specific distance matrices have a unified scale, which facilitates the fusion between views. Under an unsupervised scenario, it is assumed that the multiple views have equal importance and fuse multiview information equally. This chapter proposes a novel multiview clustering method based on distance fusion with conducted experiments to demonstrated the superiority of the method. The proposed clustering method is a partitional clustering algorithm. It requests the number of clusters to be predefined. In addition, the influence of different forms of distance fusion and the resulting distance matrices are analyzed.

Chapter 6

Dual Distance Adaptive Multiview Clustering

This chapter explores extreme learning machine as an embedding learning method for multiview clustering. Section 6.1 reviews the existing unsupervised learning methods that employ extreme learning machine for embedding learning. Section 6.2 describes the proposed method for multiview clustering. Section 6.3 provides the results of empirical comparative analysis of the proposed method and existing multiview clustering methods from different angles.

6.1 Background and Motivation

By using the weighted sum of nonlinear activation functions, ELM is able to achieve nonlinear mapping from original space to embedding space. It is proven in the previous work [67] that the distance measured in ELM embedding space makes the graph more reliable in single-view clustering. Multi-view learning with adaptive neighbors (MLAN) [104] learns the local structure of multiview data and the cluster indicator simultaneously. During the learning of the global graph, the weight for each view is adjusted automatically. With constrained Laplacian rank, the optimal graph has the number of components equals the predefined number of clusters. In light of this, this chapter extends the idea to multiview clustering and proposes dual distances adaptive multiview clustering (DAMC). The proposed method first learns an ELM-based embedding space for each view and measures the pairwise embedding distances. The dual distance is the combination of both distances measured in original space and embedding space. Similar to MLAN, the proposed approach constructs a common graph based on the weighted sum of the view-specific dual distances and adaptively adjusts the weights. We also optimize the graph with a constraint on the rank of its Laplacian. This ensures the graph has a desired number of components to serve the multiview clustering task.

On the other hand, k-means clustering based on the similarity resides on the original data space is not effective for many real applications. According to the smoothness assumption [152] in machine learning, if two input data points are close in the data space then their outputs are likely to be close. Therefore, many methods [64, 134] learn an ambient space with higher dimension while maintaining the intrinsic geometry of the data space, and applied k-means clustering in the learned ambient space. One of the reason of clustering data in the embedding space is that many real world data patterns are not linearly separable in the original space (e.g. a circle contains another circle), while they can be separated in the embedding space that is nonlinear combination of the original space basis [64, 153]. To preserve the intrinsic structure of the original data in ambient space, manifold regularization [135] is proposed as a type of regularization that uses the geometry structure of the dataset to constraint the embedding. Particularly, to avoid overfitting, it uses the Laplacian norm of the data to estimate the marginal distribution which should be large where the target function is smooth.

6.2 Proposed Method

The goal is to obtain a better distance metric to improve multiview clustering performance. Rather than learning the similarity based on the distances measured in original spaces or linearly projected spaces, we combine the existing distances with distances measured in the non-linearly mapped spaces. We aim to explore different ways to combine distance metrics and find the one that can enhance the resultant similarity matrix. Such a combination can produce a dual distance metric that measures both the explicit and implicit pairwise relationships and can be applied to the data objects lie on the multiple complex manifolds.

6.2.1 Dual Distance

The squared Euclidean distance between a pair of data, $\mathbf{x}_i$ and $\mathbf{x}_j$, is defined as:

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \quad (6.1)$$

The problem of this distance metric is that the scales are different for the distances measured from different spaces. The one with larger scale will dominate over the others during the information fusion. Intuitively, this can be simply solved by normalization. Yet, another problem is, the pairwise distance can be different when it is seen from the perspective of different data samples. In other words, the distance matrix can be converted to two different forms via row normalization and column normalization.

In this work, we fuse these two normalized matrices by taking the geometric mean of them. Hence, for two data samples in a specific space, the normalized pairwise distance is defined as:

$$\bar{d}_{ij} = \sqrt{\frac{d_{ij}}{\|\mathbf{d}_i\|_2} \cdot \frac{d_{ij}}{\|\mathbf{d}_j\|_2}} \quad (6.2)$$

where $\mathbf{d}_i$ contains all the distances linked i th data sample. This distance contains the ranking information from two perspectives and ensures a unified scale for all spaces.

6.2.1.1 ELM Embedding Space

Although the normalized distance provides a comprehensive dissimilarity measure, it may still be limited by the features. As mentioned, the actual data structure may not lie on the original feature space but a more complex manifold. Regarding this, we enhance this distance metric by combining it with another distance metric measured in the ELM embedding space.

For each view, same as ELM-CLR [67], the model is trained based on the data X and the corresponding affinity matrix A . We use the ELM output space as the embedding space. The features of data in the embedding space are the output of ELM $\mathbf{H}\boldsymbol{\beta}$. The desire embedding space preserves the pairwise similarity, and the features in the embedding space are uncorrelated. Thus, we train the ELM by solving the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \sum_{i,j}^n ||\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{h}(\mathbf{x}_j)\boldsymbol{\beta}||a_{ij} + \delta ||\boldsymbol{\beta}||_2^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^T \bar{H}^T \bar{H} \boldsymbol{\beta} = I \end{aligned} \quad (6.3)$$

where $\mathbf{h}(\mathbf{x}_i)$ is the hidden output of the data sample $\mathbf{x}_i$, $\boldsymbol{\beta}$ is the output weights to be learned, $\bar{H} = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)H$ is the averaged centered version of the hidden layer output H . The first term adjusts the pairwise distance in the output space based on the corresponding pairwise similarity a_{ij} , where a larger similarity leads to a small distance. A regularization term is formulated to prevent overfitting, and δ is the predefined trade-off parameter. To avoid the trivial solution and obtain uncorrelated features, the covariance matrix of the ELM output is constrained to be an identity matrix. Depend on the number of data samples and the number of hidden neurons, there are two different cases: 1. *the number of data samples is larger than the number of hidden neurons* ($n \geq r$).

We transform Eq.(6.3) into matrix form:

$$\begin{aligned} & \sum_{i,j}^n ||\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{h}(\mathbf{x}_j)\boldsymbol{\beta}||a_{ij} + \delta ||\boldsymbol{\beta}||_2^2 \\ = & \text{tr}(\boldsymbol{\beta}^T H^T L_A H \boldsymbol{\beta}) + \delta \text{tr}(\boldsymbol{\beta}^T \boldsymbol{\beta}) \\ = & \text{tr}(\boldsymbol{\beta}^T H^T L_A H \boldsymbol{\beta} + \delta \boldsymbol{\beta}^T \boldsymbol{\beta}) \\ = & \text{tr}(\boldsymbol{\beta}^T (H^T L_A H + \delta I) \boldsymbol{\beta}) \end{aligned} \quad (6.4)$$

where L_A is the Laplacian of the affinity matrix A , the problem can be solved via eigendecomposition. Take into account the constraint $\beta^T \bar{H}^T \bar{H} \beta = I$, assume the ELM output space has o dimensions, the optimal β is formed by the eigenvectors corresponding to the o smallest generalized eigenvalues of the problem:

$$(H^T L_A H + \delta I) \mu = \tau \bar{H}^T \bar{H} \mu$$

Each eigenvector is normalized as:

$$\bar{\mu} = \frac{\mu}{\|\bar{H} \mu\|_2}$$

and the optimal output weights is formed by the o smallest normalized eigenvectors as:

$$\beta = [\bar{\mu}_1, \bar{\mu}_2, \dots, \bar{\mu}_o] \quad (6.5)$$

2. the number of data samples is less than the number of hidden neurons ($n < r$).

Because the optimal $\beta \in \mathbb{R}^{r \times o}$ is underdetermined in this case, we aim at a smaller matrix $N \in \mathbb{R}^{n \times o}$ with $\beta = H^T N$. By substitution, the matrix form of Eq.(6.4) becomes:

$$\begin{aligned} & tr(\beta^T (H^T L_A H + \delta I) \beta) \\ = & tr(N^T H (H^T L_A H + \delta I) H^T N) \\ = & tr(N^T (H H^T L_A H H^T + \delta H H^T) N) \\ = & tr(N^T (H H^T (L_A H H^T + \delta I)) N) \end{aligned} \quad (6.6)$$

The constraint becomes $N^T H \bar{H}^T \bar{H} H^T N = I$, and the optimal N can be obtained by solving the following generalized eigenvalue problem:

$$\begin{aligned} & H H^T (H^T L_A H + \delta I) \mu = \tau H \bar{H}^T \bar{H} H^T \mu \\ & (H^T L_A H + \delta I) \mu = \tau (H H^T)^{-1} H \bar{H}^T \bar{H} H^T \mu \end{aligned}$$

In this case, each eigenvector is normalized as:

$$\bar{\mu} = \frac{\mu}{\|\bar{H} H^T \mu\|_2}$$

and the optimal solution of the output weights is formed as:

$$\beta = H^T[\bar{\mu}_1, \bar{\mu}_2, \dots, \bar{\mu}_o] \quad (6.7)$$

6.2.1.2 Combination of distance metrics

With the trained output weights β , we now use ELM to enhance the distance metrics. Specifically, the original data is fed into extreme learning machine and transferred to the embedding space. Then, the pairwise normalized distances in the embedding space are measured and combined with those measured in the original space. The pairwise distance in the embedding space is computed as:

$$d_{ij,ELM} = \|\mathbf{h}(\mathbf{x}_i)\beta - \mathbf{h}(\mathbf{x}_j)\beta\|_2^2 \quad (6.8)$$

where $\mathbf{h}(\mathbf{x}_i)$ is the output of hidden layer and β is the output weights. For convenience, we call the distance measured in the embedding space as embedding distance and it is not the distance between embedding spaces. Of course, the embedding distance is also normalized in the way as stated in Eq.(6.2). We aim to utilize both the original distance and the embedding distance to construct a more comprehensive graph. Therefore, we fuse the distances into a dual distance metric as:

$$D_d = f(D_o, D_e) \quad (6.9)$$

where D_o is the original distance matrix, D_e is the embedding distance matrix. We use a symmetric function as the combination function so that the two distances equally contribute to the dual distance. The candidates of the combination function are the symmetric element-wise functions: geometric mean, arithmetic mean, sum, product. It is worth noting that, different from ELM-CLR, the embedding spaces are fixed after training. We empirically analysis these combination functions in the experiments.

6.2.2 Dual Distance Adaptive Multiview Clustering

After defining the distance metrics, we now introduce the proposed dual distance adaptive multiview clustering. The proposed method optimizes a global affinity

matrix for all the views in following formulation:

$$\begin{aligned}
& \min_{A \in \mathbb{R}^{n \times n}} \sum_i^v \sqrt{\text{tr}(D_{d,i}^T A)} + \alpha \|A\|_F^2 \\
& \text{s.t.} \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1, \\
& \quad \text{rank}(L_A) = n - c
\end{aligned} \tag{6.10}$$

where D_e^T is the dual distance matrix of the i th view computed via Eq.(6.9), A is the common affinity matrix, $\text{tr}(\cdot)$ indicates the trace of matrix, α is the regularization parameter. The first term learns the similarities from the pairwise distances by minimizing the sum of the element-wise products of the distance matrix and the affinity matrix, so that the smaller distances lead to higher similarities. The rank of the Laplacian is constrained by the number of data samples n and the desired number of clusters c . This constraint ensures that there is exact c components in the graph. Considering the constraints, we write the Lagrange function of Eq.(6.10) as:

$$\sum_i^v \sqrt{\text{tr}(D_{d,i}^T A)} + \alpha \|A\|_F^2 + \mathcal{G}(\Lambda, A) \tag{6.11}$$

where Λ the Lagrangian multiplier, $\mathcal{G}(\Lambda, A)$ contains the terms derived from the constraints on the affinity matrix A . Similar to MLAN, we define the weight of the i th view as:

$$w_i = (2\sqrt{\text{tr}(D_{d,i}^T A)})^{-1} \tag{6.12}$$

with Eq.(6.12), let the derivative of Eq.(6.14) with respect to A equal to zero, we have:

$$\sum_i^v w_i \frac{\partial \text{tr}(D_{d,i}^T A)}{\partial A} + \alpha \frac{\partial \|A\|_F^2}{\partial A} + \frac{\partial \mathcal{G}(\Lambda, A)}{\partial A} \tag{6.13}$$

the optimization problem can be considered as:

$$\begin{aligned}
& \min_{A \in \mathbb{R}^{n \times n}} \sum_i^v w_i \text{tr}(D_{d,i}^T A) + \alpha \|A\|_F^2 \\
& \text{s.t.} \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1, \\
& \quad \text{rank}(L_A) = n - c
\end{aligned} \tag{6.14}$$

Notice that we fix the distance matrices, so the weights $\mathbf{w}$ solely depend on the affinity matrix A . Thus, A and $\mathbf{w}$ can be optimized via alternating optimization. When A converges, we use Tarjan's strongly connected components algorithm [22]

to find the connected components in A . There will be exact c connected components in the graph and each components represents a cluster. The proposed DAMC is summarized in Algorithm 1.

Algorithm 3: DAMC

Require: Observed multiview data $\mathcal{X} = \{X_1, X_2, \dots, X_v\}$,
number of clusters c ,
number of nearest neighbours k .
Ensure: Affinity matrix $A \in \mathbb{R}^{n \times n}$ with exact c connected components
for each view **do**
 Compute the normalized original distances $\bar{D}_o$ via Eq.(6.1) and Eq.(6.2);
 Initialize the affinity matrix A via Eq.(6.28);
 Compute the output weights β via Eq.(6.5) or Eq.(6.7);
 Compute the normalized embedding distances $\bar{D}_e$ via Eq.(6.8) and Eq.(6.2);
 Compute the dual distance D_d via Eq.(6.9)
end for
repeat
 Update F by solving Eq.(6.17);
 Update $\mathbf{w}$ via Eq.(6.12);
 Update A via Eq.(6.28);
until the stopping criterion

6.2.3 Optimization

We adopt alternating optimization to solve the formulated problem. After the initialization, we first update the affinity matrix A with stationary weights $\mathbf{w}$, then update $\mathbf{w}$ with fixed A . Whenever A is updated, the eigenvectors of the Laplacian of A is also updated. We repeat the process until the Laplacian rank constraint is ensured. Thus, we start by converting the constraint to an optimizable term and embed it into the formulation. Since the Laplacian L_A is positive semi-definite, all the eigenvalues of L_A are greater or equal to zero. Let $\sigma_i(L_A)$ denote the i th smallest eigenvalue of L_A , the Laplacian rank constraint $\text{rank}(L_A) = n - c$ can be guaranteed by setting all the c smallest eigenvalues to zero:

$$\sum_{i=1}^c \sigma_i(L_A) = 0$$

which, according to Ky Fan's Theorem [142], can be achieved by:

$$\min_{F \in \mathbb{R}^{n \times c}, F^T F = I} \text{tr}(F^T L_A F) \quad (6.15)$$

Replacing the Laplacian rank constraint in Eq.(6.14) with the optimizable term, the problem becomes:

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} & \sum_i^v w_i \text{tr}(D_{d,i}^T A) + \alpha \|A\|_F^2 + 2\lambda \text{tr}(F^T L_A F) \\ \text{s.t.} & \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (6.16)$$

where λ is the parameter with a large value to ensure $\sum_{i=1}^c \sigma_i(L_A) = 0$ so that the constraint $\text{rank}(L_A) = n - c$ is satisfied. Since F only depends on the affinity matrix A , F and $\mathbf{w}$ can be updated at the same time.

6.2.3.1 Update $\mathbf{w}$ and F

To update $\mathbf{w}$ and F , we fix the affinity matrix A in Eq.(6.16). The weights $\mathbf{w}$ can be simply updated via Eq.(6.12). Thus, the first two terms of Eq.(6.14) are constants. The remaining part of the problem become:

$$\min_{F \in \mathbb{R}^{n \times c}, F^T F = I} \text{tr}(F^T L_A F) \quad (6.17)$$

The optimal F is derived via eigendecomposition, where the columns of F are the eigenvectors corresponding to the c smallest eigenvalues of L_A .

6.2.3.2 Update A

When update A , $\mathbf{w}$ and F have to be fixed. Eq.(6.16) can be rewritten as:

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} & \text{tr}\left(\sum_i^v w_i D_{d,i}^T A + \alpha A^T A + 2\lambda F^T L_A F\right) \\ \text{s.t.} & \quad \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (6.18)$$

Since the weights are fixed, all the distance matrices can be fused into a single dual distance matrix denoted as D_d . The problem becomes:

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times n}} \quad & tr(D_d^T A + \alpha A^T A + 2\lambda F^T L_A F) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (6.19)$$

With the elementary equation in spectral analysis

$$\sum_{i,j}^n \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 a_{ij} = 2tr(F^T L_A F)$$

we denote $d_{f,ij} = \|\mathbf{f}_i - \mathbf{f}_j\|_2^2$ and rewrite the problem in scalar form:

$$\begin{aligned} \min_{a_{ij}} \quad & \sum_{i,j}^n (d_{d,ij} a_{ij} + \alpha a_{ij}^2 + \lambda d_{f,ij} a_{ij}) \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (6.20)$$

Notice that the constraint $\mathbf{a}_i^T \mathbf{1} = 1$ is applied to the rows, so each column of the affinity matrix A can be optimized separately. Denote $\hat{\mathbf{d}}_i = \mathbf{d}_{d,i} + \lambda \mathbf{d}_{f,i}$, we have:

$$\begin{aligned} \min_{\mathbf{a}_i} \quad & \|\mathbf{a}_i + \frac{1}{2\alpha} \hat{\mathbf{d}}_i\|_2^2 \\ \text{s.t.} \quad & \forall i, \mathbf{a}_i^T \mathbf{1} = 1, 0 \leq a_{ij} \leq 1 \end{aligned} \quad (6.21)$$

where the parameter α is determined during the initialization. The Lagrange function of Eq.(6.21) is written as:

$$\mathcal{L}(\mathbf{a}_i, \zeta, \boldsymbol{\xi}_i) = \frac{1}{2} \|\mathbf{a}_i + \frac{\hat{\mathbf{d}}_i}{2\alpha}\|_2^2 - \zeta(\mathbf{a}_i^T \mathbf{1} - 1) - \boldsymbol{\xi}_i^T \mathbf{a}_i \quad (6.22)$$

where ζ and $\boldsymbol{\xi}_i$ are the Lagrangian multipliers. According to the KKT condition, $\mathbf{a}_i$ can be optimized as

$$a_{ij} = \max(0, \zeta - \frac{\hat{d}_{ij}}{2\alpha}) \quad (6.23)$$

According to CAN, ζ is computed as:

$$\zeta = \frac{1}{k} + \frac{1}{2k\alpha} \sum_{j=1}^k \hat{d}_{ij}$$

where k is the predefined number of nearest neighbours. With the optimized $\mathbf{a}_i$, we are able to determine the parameter α .

6.2.3.3 Parameter Determination

α is determined before the alternating optimization, where A is computed based on the dual distance D_d without the Laplacian rank constraint. Therefore, $d_{f,ij}$ is not involved in Eq.(6.20) and $\hat{d}_i = d_{d,i}$. Denote $\hat{d}_{i,k}$ and $a_{i,k}$ as the k th smallest elements in $\hat{\mathbf{d}}_i$ and $\mathbf{a}_i$. To ensure there are exact k neighbours for each of the n data samples in the affinity matrix A , we have:

$$\begin{cases} a_{i,k} & > 0 \\ a_{i,k+1} & = 0 \end{cases} \quad (6.24)$$

Substitute Eq.(6.23) into Eq.(6.24), we have

$$\begin{cases} \zeta - \hat{d}_{i,k}/2\alpha & > 0 \\ \zeta - \hat{d}_{i,k+1}/2\alpha & \leq 0 \end{cases} \quad (6.25)$$

Based on the constraint $\mathbf{a}_i^T \mathbf{1} = 1$, we have

$$\begin{aligned} & 1 \sum_{j=1}^k (\zeta - d_{i,k}/2\alpha) \\ \Rightarrow \zeta &= \frac{1}{k} + \frac{1}{2k\alpha} \sum_{j=1}^k \hat{d}_{ij} \end{aligned} \quad (6.26)$$

Substitute Eq.(6.26) into Eq.(6.25), we have

$$\begin{cases} \frac{k}{2} \hat{d}_{i,k+1} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} & \geq \alpha \\ \frac{k}{2} \hat{d}_{i,k} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} & < \alpha \end{cases} \quad (6.27)$$

To make the optimal solution of $\mathbf{a}_i$ having exact k nonnegative elements, α is set to the maximal value:

$$\alpha = \frac{k}{2} \hat{d}_{i,k+1} - \frac{1}{2} \sum_{j=1}^k \hat{d}_{ij} \quad (6.28)$$

Obviously, each $\hat{\mathbf{d}}_i$ has a specific α , thus the final α is computed as the arithmetic mean of all the candidates.

6.2.3.4 Stopping Criterion

The parameter λ is not fixed to a predefined value, because it is hard to determine the optimal λ for each datasets beforehand and unsuitable λ might lead to undesired number of clusters. Therefore, λ is updated during the alternating optimization, the algorithm will stop when λ stops updating. Specifically, λ is initialized to α and updated according to the eigenvalues. If the sum of the eigenvectors corresponding to the smallest c eigenvalues is greater than a tiny threshold, λ is doubled. If the sum of the eigenvectors corresponding to the smallest $c + 1$ eigenvalues is less than the predefined threshold, we cut λ in half. The above updating strategy sets $\lambda = \alpha * 2^t$ and adjusts λ by tuning the integer t . If we keep increasing λ with a small step, we will find the boundary where c -th eigenvalue reduced from a small positive value to zero. The boundary condition leads to a larger $c + 1$ -th eigenvalue, but the influence of a larger $c + 1$ -th eigenvalue to the performance is unknown. Finding such λ could be time consuming. Thus, instead of finding the boundary condition, this updating strategy looks for a suitable λ satisfying the constraint. In practice, this strategy speeds up the procedure and determines suitable λ much faster than grid search. λ and the eigenvalues in every iterations are plotted in Fig. 6.1. In most of the cases, with λ increasing, the c -th smallest eigenvalue vanishes and the $(c + 1)$ -th smallest eigenvalue remain nonzero. Meanwhile, this updating strategy can prevent the vanishment of $(c + 1)$ -th smallest eigenvalue. In the experiment on HD dataset, the $(c + 1)$ -th smallest eigenvalue vanished with the c -th smallest eigenvalue in 12-th iteration. Accordingly, the value of λ is reduced by half. In the next iteration, we obtain a nonzero $(c + 1)$ -th smallest eigenvalue and a c -th smallest eigenvalue close to zero.

FIGURE 6.1: λ is updated in each iteration until the c -th smallest eigenvalue of the Laplacian vanished

6.3 Experiments

This section conducts experiments to evaluate the performance of the proposed method and the state-of-the-art multiview clustering methods on various public multiview datasets.

6.3.1 Datasets

Handwritten Digits (HD) [145], MSRCv1, and COIL20 [147] are described in the previous chapter and they are adopted in this section. Caltech101 is a digital image dataset consists of pictures of objects belonging to 101 distinct object categories. Following GSF [105], this thesis adopt the widely used 7 classes and 1474 images. The 7 classes are faces, motorbikes, dollar bill, Garfield, snoopy, stop sign, Windsor chair. Six feature sets are extracted from the images: Gabor with 48 dimensions, wavelet-moment with 40 dimensions, CENTRIST with 254 dimensions, histogram of oriented gradients (HOG) with 1948 dimensions, GIST with 512 dimensions, and local binary patterns (LBP) with 928 dimensions.

FIGURE 6.2: Samples of Handwritten Digits Dataset

6.3.2 Algorithms and Settings

For comparison, we evaluate the proposed DAMC along with existing state-of-the-art multiview clustering methods. Co-regularized multi-view spectral clustering (CMSC) [103], Multi-modal spectral clustering (MMSC) [149], Multiview consensus

graph clustering (MCGC)[72], and Multi-view learning with adaptive neighbours (MLAN) [104] are described in the previous chapter and they are adopted in this section.

DFAN: Distance fusion with adaptive neighbors described in previous chapter uses distance fusion to obtain unified measure of pairwise distance. Such measures from different views can be combine with equal weights. The multiview clustering problem is then converted to a single-view clustering problem.

When constructing the KNN graphs, the number of nearest neighbours is picked from $\{3, 4, 5, 6, 7, 8, 9, 10\}$. For the predefined hyperparameters, this thesis uses the same settings in the original papers.

The co-regularization parameter λ of CMSC is in the range from 0.01 to 0.05 with step 0.01. For MMSC, the penalty parameter is set as $\log_{10} \alpha$ with α in the range from -2 to 2 with incremental step 0.2. The parameter β of MCGC is set to 0.6, β is picked from a larger range $\{0.01, 0.1, 0.6, 1, 10, 100\}$. We conduct each experiment for 10 times and take the average, the best results are reported.

6.3.3 Evaluation Metrics

The clustering accuracy is used as the performance metric to evaluate and compare the methods:

$$ACC = \frac{\sum_{i=1}^n \delta(y_i, \text{map}(y_i^*))}{n} \quad (6.29)$$

and

$$\delta(y_i, \text{map}(y_i^*)) = \begin{cases} 1 & y_i = y_i^* \\ 0 & \text{otherwise} \end{cases} \quad (6.30)$$

where $\text{map}(\cdot)$ is the best mapping function that maximizes the accuracy by Hungarian algorithm [151], which maps a predicted label to a true cluster label. y_i and y_i^* is the true cluster label and predicted cluster label of the i th sample, and n is the total number of data samples.

The second performance metric is a normalized version of mutual information. For predicted label Y and ground truth Y^* , the metric can be computed as:

$$NMI(Y, Y^*) = \frac{H(Y) - H(Y|Y^*)}{\sqrt{H(Y) * H(Y^*)}} \quad (6.31)$$

Purity is used as another measure of clustering performance, which can be defined as:

$$Purity(Y, Y^*) = \frac{1}{n} \sum_{m \in c} \max_{d \in c^*} |m \cap d| \quad (6.32)$$

where n is the number of data samples, c is the number of clusters in Y , c^* is the number of classes in Y^* .

An alternative interpretation of clustering is to view it as $n(n-1)/2$ pairwise decisions. For a pair of samples have the same ground truth label, their pairwise decision is true positive (TP) if they are grouped in the same cluster. Otherwise, the decision is false negative (FN). For a pair of samples have different ground truth label, their pairwise decision is false positive (FP) if they are grouped in the same cluster. Otherwise, the decision is true negative (TN). The precision and recall are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (6.33)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6.34)$$

and the default balanced F-score is

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6.35)$$

Rand index is the percentage of correct decisions among all decisions:

$$\text{Rand index} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (6.36)$$

The adjusted Rand index (ARI) is the corrected-for-chance version of the Rand index, which is defined as

$$\text{ARI} = \frac{\sum_{ij} C_{n_{ij}}^2 - (\sum_i C_{a_i}^2 \sum_j C_{b_j}^2) / C_n^2}{(\sum_i C_{a_i}^2 + \sum_j C_{b_j}^2) / 2 - (\sum_i C_{a_i}^2 \sum_j C_{b_j}^2) / C_n^2} \quad (6.37)$$

where C_n^m is the combination operation, which selects m items from a collection of number n . n_{ij} , a_i , b_j are values from the contingency table.

6.3.4 Comparison with Existing methods

This section lists the experimental results of all the tested algorithms in Table 6.1 to compare their multiview clustering performance. For each dataset, the highest results in bold and the values inside the brackets are the corresponding standard deviations.

Obviously, the proposed approach achieves the highest results in most cases. In the rest cases, DAMC also offers competitive multiview clustering performance. Because the proposed approach is closely related to MLAN, this thesis is more concerned about surpassing the performance of it. Compared with MLAN, DAMC has higher scores in all the metrics. These are convincing evidences that, the dual distance is more reliable than the Euclidean distance measured in the original data space. With the dual distance, DAMC is more effective than the state-of-the-art MLAN in multiview clustering problem.

It is interesting that DAMC tends to lose in recall, especially when compared with MCGC. Yet, MCGC usually has very low precision and loses to DAMC. Following is the F-score which considers both precision and recall, DAMC obtains the highest F-score on all the datasets. It can be concluded as DAMC maintains a good balance between precision and recall.

CMSC produces the worst results with a large standard deviation in most cases. The reason can be found in the differences between CMSC and the rest. It is the only one fuses the multiple cluster indicators after the clustering where most information has been lost. The rest methods fuse multiview information before the eigendecomposition of spectral clustering. Thus, the timing of fusion is a very important factor. The key is to avoid losing information before fusion.

With respect to DFAN, DAMC performs better in MSRCv1 and almost the same in HD and COIL20. In HD and COIL20, the samples from the same classes are alike to each other. While in MSRCv1, the samples of the same classes are much more different and contain more noise. We can conclude that dual distance is more robust to noise thanks to the embedding space generated by extreme learning machine.

To summarize, DAMC can achieve at least comparable performance when compared with state-of-the-art multiview clustering methods. DAMC surpasses MLAN in all experiments, this reflects the effectiveness of the proposed dual distance.

FIGURE 6.3: Samples of MSRCv1 Dataset

FIGURE 6.4: Samples of COIL20 Dataset

TABLE 6.1: Comparison of Multiview Clustering Algorithms (%)

		CMSC	MMSC	MCGC	MLAN	DFAN	DAMC (proposed)
MSRCv1	ACC	68.90(+10.80)	77.62(+4.67)	72.86(+0.00)	81.90(+0.00)	82.86(+0.00)	84.29(+0.00)
	NMI	62.96(+6.21)	72.79(+1.75)	66.13(+0.00)	77.02(+0.00)	78.05(+0.00)	80.58(+0.36)
	PUR	72.19(+8.63)	80.62(+3.17)	76.19(+0.00)	81.90(+0.00)	82.86(+0.00)	84.29(+0.00)
	PRE	57.62(+8.50)	66.19(+1.17)	61.60(+0.00)	70.10(+0.00)	69.67(+0.00)	72.00(+0.00)
	REC	61.18(+8.83)	72.51(+3.83)	94.38(+0.00)	77.60(+0.00)	82.40(+0.00)	86.54(+0.00)
	FSC	59.33(+9.14)	68.94(+4.83)	63.90(+0.00)	73.66(+0.00)	75.25(+0.00)	76.87(+0.00)
	ARI	52.52(+10.42)	63.67(+4.55)	57.82(+0.00)	69.16(+0.00)	70.86(+0.00)	72.77(+0.00)
HD	ACC	87.00(+7.38)	93.98(+5.77)	95.85(+0.00)	97.40(+0.00)	98.10(+0.00)	98.05(+0.00)
	NMI	82.06(+4.73)	93.60(+2.28)	91.36(+0.00)	94.29(+0.00)	95.59(+0.00)	95.63(+0.03)
	PUR	87.50(+6.61)	94.49(+4.95)	95.85(+0.00)	97.40(+0.00)	98.10(+0.00)	98.05(+0.00)
	PRE	79.01(+4.54)	90.28(+2.70)	91.71(+0.00)	94.78(+0.27)	96.13(+0.00)	96.09(+0.34)
	REC	80.98(+7.11)	94.16(+5.24)	99.12(+0.00)	95.21(+0.00)	98.02(+0.00)	98.12(+0.00)
	FSC	79.96(+7.97)	92.00(+7.94)	91.95(+0.00)	94.85(+0.00)	96.19(+0.00)	96.14(+0.00)
	ARI	77.70(+7.95)	91.07(+5.89)	91.05(+0.00)	94.28(+0.00)	95.77(+0.00)	95.71(+0.00)
COIL20	ACC	71.58(+4.41)	76.03(+6.34)	73.82(+0.00)	87.50(+0.00)	99.86(+0.00)	99.65(+0.00)
	NMI	83.17(+1.27)	91.65(+2.07)	86.89(+0.00)	96.64(+0.00)	99.79(+0.00)	99.50(+0.00)
	PUR	74.04(+3.11)	80.20(+4.98)	78.06(+0.00)	89.86(+0.00)	99.86(+0.00)	99.65(+0.00)
	PRE	66.77(+2.35)	67.69(+3.73)	63.13(+0.00)	82.29(+0.00)	99.72(+0.00)	99.30(+0.00)
	REC	72.24(+2.82)	89.53(+5.02)	81.10(+0.00)	96.83(+0.00)	99.73(+0.00)	99.73(+0.00)
	FSC	69.35(+3.92)	75.27(+3.64)	70.15(+0.00)	88.97(+0.00)	99.72(+0.00)	99.31(+0.00)
	ARI	67.70(+3.00)	73.76(+5.37)	68.40(+0.00)	88.35(+0.00)	99.71(+0.00)	99.27(+0.00)
Caltech101	ACC	44.40(+3.52)	67.50(+0.85)	65.06(+0.00)	79.58(+0.00)	82.23(+0.00)	83.04(+1.29)
	NMI	47.23(+0.97)	64.32(+1.43)	55.08(+0.00)	60.24(+0.00)	61.40(+0.00)	62.48(+4.91)
	PUR	81.60(+1.77)	86.91(+1.37)	87.79(+0.00)	85.75(+0.00)	83.51(+0.00)	85.93(+0.73)
	PRE	81.89(+1.02)	89.40(+3.10)	86.13(+0.00)	68.31(+0.00)	72.72(+0.00)	86.17(+0.02)
	REC	33.38(+1.37)	58.76(+1.38)	99.22(+0.00)	88.98(+0.00)	98.28(+0.00)	98.62(+3.92)
	FSC	47.42(+2.49)	70.37(+1.11)	63.73(+0.00)	75.16(+0.00)	81.14(+0.00)	83.20(+0.26)
	ARI	32.31(+1.74)	57.47(+1.70)	44.17(+0.00)	56.55(+0.00)	66.90(+0.00)	71.34(+7.63)

TABLE 6.2: Comparison of Combination Functions (%)

		geometric mean	product	arithmetic mean	sum	MLAN
MSRCv1	ACC	84.29(+0.00)	83.33(+0.00)	82.54(+1.37)	83.33(+0.00)	81.90(+0.00)
	NMI	79.78(+0.20)	78.85(+0.00)	80.58(+0.36)	79.89(+0.00)	77.02(+0.00)
	PUR	84.29(+0.00)	83.33(+0.00)	83.02(+0.55)	83.33(+0.00)	81.90(+0.00)
	PRE	72.00(+0.00)	70.08(+0.00)	70.00(+0.76)	70.74(+0.00)	70.10(+0.00)
	REC	86.54(+0.00)	83.12(+0.00)	83.19(+0.00)	84.17(+0.00)	77.60(+0.00)
	FSC	76.80(+0.13)	74.53(+0.00)	75.23(+0.87)	76.87(+0.00)	73.66(+0.00)
	ARI	72.68(+0.15)	70.02(+0.00)	70.88(+1.02)	72.77(+0.00)	69.16(+0.00)
HD	ACC	94.28(+5.40)	98.05(+0.00)	97.55(+0.17)	94.38(+5.40)	97.40(+0.00)
	NMI	93.60(+1.25)	95.63(+0.03)	94.57(+0.27)	93.57(+1.52)	94.29(+0.00)
	PUR	94.32(+5.34)	98.05(+0.00)	97.55(+0.17)	94.42(+5.34)	97.40(+0.00)
	PRE	89.61(+9.00)	96.09(+0.00)	95.10(+0.33)	90.11(+0.05)	94.78(+0.27)
	REC	98.12(+0.34)	97.54(+0.25)	98.12(+0.00)	98.19(+4.48)	95.21(+0.00)
	FSC	92.19(+4.64)	96.14(+0.00)	95.15(+0.33)	92.48(+8.49)	94.85(+0.00)
	ARI	91.27(+5.24)	95.71(+0.00)	94.62(+0.37)	91.61(+5.05)	94.28(+0.00)
COIL20	ACC	97.57(+0.49)	97.31(+0.04)	99.65(+0.00)	98.87(+1.36)	87.50(+0.00)
	NMI	98.73(+0.00)	98.46(+0.06)	99.50(+0.00)	99.13(+0.64)	96.64(+0.00)
	PUR	97.57(+0.49)	97.31(+0.04)	99.65(+0.00)	98.87(+1.36)	89.86(+0.00)
	PRE	95.05(+0.93)	95.36(+0.16)	99.30(+0.00)	97.75(+0.00)	82.29(+0.00)
	REC	99.73(+0.00)	98.84(+0.00)	99.32(+0.00)	99.46(+1.97)	96.83(+0.00)
	FSC	96.39(+2.85)	95.90(+0.00)	99.31(+0.00)	98.17(+2.68)	88.97(+0.00)
	ARI	96.20(+3.00)	95.69(+0.00)	99.27(+0.00)	98.08(+2.07)	88.35(+0.00)
Caltech	ACC	81.80(+0.21)	83.04(+1.29)	81.93(+0.79)	81.03(+0.63)	79.58(+0.00)
	NMI	60.41(+1.07)	62.48(+4.91)	61.82(+4.41)	58.04(+1.37)	60.24(+0.00)
	PUR	83.76(+0.04)	85.93(+0.73)	83.72(+0.00)	83.08(+0.20)	85.75(+0.00)
	PRE	74.45(+0.19)	86.17(+0.26)	78.61(+8.59)	72.76(+0.21)	68.31(+0.00)
	REC	97.23(+0.59)	96.59(+0.00)	98.62(+0.02)	98.58(+0.73)	88.98(+0.00)
	FSC	81.41(+0.49)	81.81(+0.00)	83.20(+3.92)	80.40(+1.15)	75.16(+0.00)
	ARI	67.83(+0.84)	68.51(+0.00)	71.34(+7.63)	65.85(+1.44)	56.55(+0.00)

TABLE 6.3: Single-view Clustering Performance (%)

		Glass	Vehicle	UMIST	warpPIE10P	YaleB
DAMC	ACC	54.91(± 2.31)	45.04(± 0.00)	81.74(± 1.23)	80.71(± 2.36)	42.56(± 1.55)
	NMI	45.97(± 0.50)	27.84(± 0.08)	92.48(± 0.78)	79.41(± 2.27)	59.27(± 1.00)
	PUR	65.42(± 0.00)	45.74(± 0.00)	84.96(± 0.37)	80.71(± 2.36)	45.42(± 0.91)
UFS-ELM	ACC	50.00(± 0.00)	51.98(± 0.06)	76.74(± 1.33)	56.19(± 0.00)	31.59(± 1.79)
	NMI	33.32(± 0.00)	21.98(± 0.36)	87.58(± 1.50)	64.91(± 0.00)	36.73(± 0.94)
	PUR	63.55(± 0.00)	51.48(± 0.30)	81.04(± 2.45)	62.86(± 0.00)	31.78(± 1.34)

6.3.5 Performance on Single-view Datasets

Although the proposed DAMC is a multiview clustering method, it can be applied to single-view data. We adopt the datasets in Chapter 3 to analysis DAMC. The results are listed in Table 6.3. In most cases, DAMC performs better than UFS-ELM demonstrating its capability on single-view data.

6.3.6 Combination function of distance matrices

As mentioned, the proposed method uses a symmetric function to combine the original distance and the embedding distance. As such, the distance metrics have equal weights to contribute to the dual distance. When evaluating DAMC, four functions are adopted and compared : geometric mean, product, arithmetic mean, sum. We list the performance of the four functions along with MLAN as the baseline in Table 6.4.

Based on the results, there is no clear evidence that can prove one of the four combination functions is the best choice, since none of them dominates in most cases. Yet, the arithmetic mean and product seem to be more reliable. They produce comparable performance even when they are not the best. Regardless of the choice of combination function, a distinct improvement in multiview clustering performance can be seen in most cases.

6.3.7 Combination of distances with unequal weights

In the proposed method, the original distance and the embedding distance are fused with equal weights. This section explores the performance of the proposed method when the distances are assigned with different weights. The experiment is done by adding the weights for the two distances and computing the dual distance as:

$$D_d = \sqrt{\gamma D_e \times (1 - \gamma) D_o} \quad (6.38)$$

where $0 < \gamma < 1$ is the predefined weight for the embedding distance and $(1 - \gamma)$ is the weight for original distance. The clustering performance is evaluated with γ increasing from 0.1 to 0.9 with a step of 0.1. The results are shown in Figure 6.5. On MSRCv1, the overall performance is stable when changing the weights. However, the most left and most right bars of recall are lower than the rest. In other words, the recall slightly drops when one of the distance is assigned with a much smaller weight. On the Handwritten Digits dataset, the performance of the fused distance stays almost the same. Here, the reason is that the fused distance remained nearly unchanged because the learned embedding distances are similar to the original distances. On COIL20 and Caltech dataset, the performance varies without a clear trend. Yet, the performance of equal weights is relatively high and more stable in all metrics. To summarize, while assigning different weights results in similar performance, the combination of distances with equal weights gives more stable performance on different datasets.

FIGURE 6.5: Performance of weighted combination of the distances

6.3.8 Analyzing the parameters

This section further analyzes the influence of the predefined parameters. The first one is the dimensionality of the embedding space which equals to the number of output neurons in ELM. The performance of DAMC with different number of outputs are shown as a bar graph in Figure 6.6. In each bar graph, the bars in the same metrics indicated by the same color are comparable in height, which means DAMC is not so sensitive to the dimensionality of the embedding space. Thanks to the distance normalization. No matter how the dimensionality changes, the resulting distances are the same. The final clustering results mainly depend on the affinity matrix, which is barely affected by this factor. Nevertheless, there is still a small advance when increasing the dimensionality to 100. The reason is that the embedding space with higher dimensionality is more discriminative.

The trade-off parameter δ is used to regularize the output weights. The bar graphs of the performance are shown in Figure 6.7. The best performances are achieved with $\delta = 10^3$ on all the datasets except for MSRCv1. The bars in the same color are relatively similar with δ picked from 10^{-3} to 10^2 . When δ is changed from 10^2 to 10^3 , the performances are suddenly improved. It is possible that the performance can be further advanced with larger value of δ .

FIGURE 6.6: Influence of the dimensionality of embedded space

FIGURE 6.7: Influence of the trade-off parameter

TABLE 6.4: Ablation Study(%)

		DAMC	DAMC(embedding distance)	DAMC(original distance)
MSRCv1	ACC	84.29(+0.00)	84.29(+0.00)	82.38(+0.00)
	NMI	80.58(+0.36)	80.93(+0.09)	75.66(+0.00)
	PUR	84.29(+0.00)	84.29(+0.00)	82.38(+0.00)
	PRE	72.00(+0.00)	72.40(+0.00)	68.97(+0.00)
	REC	86.54(+0.00)	87.98(+0.20)	80.43(+0.00)
	FSC	76.87(+0.00)	77.03(+0.32)	74.26(+0.00)
	ARI	72.77(+0.00)	73.05(+0.25)	69.74(+0.00)
HD	ACC	98.05(+0.00)	97.52(+0.11)	87.15(+0.00)
	NMI	95.63(+0.03)	94.64(+0.28)	90.80(+0.00)
	PUR	98.05(+0.00)	97.52(+0.11)	87.45(+0.00)
	PRE	96.09(+0.34)	95.04(+0.00)	78.04(+0.00)
	REC	98.19(+4.48)	98.12(+0.23)	97.84(+0.00)
	FSC	96.14(+0.00)	95.11(+0.24)	85.25(+0.00)
	ARI	95.71(+0.00)	94.57(+0.25)	83.45(+0.00)
COIL20	ACC	99.65(+0.00)	99.65(+0.00)	92.78(+0.00)
	NMI	99.50(+0.00)	99.50(+0.00)	98.00(+0.00)
	PUR	99.65(+0.00)	99.65(+0.00)	94.79(+0.00)
	PRE	99.30(+0.00)	99.30(+0.00)	90.21(+0.00)
	REC	99.73(+0.00)	99.46(+0.00)	97.33(+0.00)
	FSC	99.31(+0.00)	99.31(+0.00)	93.63(+0.00)
	ARI	99.27(+0.00)	99.27(+0.00)	93.29(+0.00)
Caltech	ACC	83.04(+1.29)	80.70(+3.41)	80.19(+0.00)
	NMI	62.48(+4.91)	61.46(+10.94)	53.97(+0.00)
	PUR	85.93(+0.73)	84.80(+1.73)	81.55(+0.00)
	PRE	86.17(+0.02)	77.32(+0.13)	62.03(+0.00)
	REC	98.62(+3.92)	97.08(+10.99)	98.63(+0.00)
	FSC	83.20(+0.26)	80.06(+18.74)	74.46(+0.00)
	ARI	71.34(+7.63)	65.77(+20.94)	52.47(+0.00)

6.3.9 Ablation study

We perform ablation study on DAMC and the results are shown as below. When removing original distance and using only embedding distance, the performances metrics drop slightly. When using only original distance, the performance drop significantly. Thus, the contributions of embedding distance and original distance are revealed.

6.4 Summary

This chapter proposes a novel multiview clustering method called DAMC. DAMC reserves several advantages of conventional spectral clustering methods, such as it is insensitive to the initialization. It produces results with desire number of clusters, and need no post-processing. Meanwhile, instead of using traditional Euclidean distance in the original data space, DAMC applies the dual distance. In DAMC, the original distances are normalized and fused with the distances measured in the embedding space. Thus, the resulting dual distances are more reliable. On various multiview datasets, DAMC demonstrates satisfactory multiview clustering performance and clear improvement to the baseline. Based on the experiments, it is robust enough to handle embedding spaces with different dimensionalities.

Chapter 7

Conclusions and Future Work

In this chapter, Section [7.1](#) concludes the works presented in Chapter 3, Chapter 4, Chapter 5, and Chapter 6. Section [7.2](#) suggests the directions for future research.

7.1 Conclusions

This thesis investigates different extreme learning machines as embedding learning methods to solve clustering problems and a distance measure for multiview fusion. Specifically, Chapter 1 and Chapter 2 introduce the background and related works regarding the extreme learning machines and clustering. Chapter 3 proposes OCA-ELM for multiview 3D shape classification. Chapter 4 proposes UFS-ELM as an end-to-end clustering approach to adopt the unsupervised feature selection method for the feature learning of the neural network. Chapter 5 proposes the distance fusions as a multiview fusion method and DFAN for multiview clustering. Chapter 6 proposes DAMC as a new multiview clustering algorithm that combines the original distances and the embedding distances for multiview clustering. The major conclusions of this thesis are listed as follows:

The optimization of neural networks often involves L1-norm or L2-norm as the regularization of the weights to avoid overfitting. The constrained-optimization-based ELM [54] minimizes the L2-norm of the output weights to maximize the distance of the separating margins between different classes in the ELM feature space. By contrast, Chapter 4 embeds the unsupervised feature selection method in the optimization, as a regularization term for the output weights. Similar to L2-norm, the adopted L2,1-norm ensures the sparsity of the output weights and enlarges the margins between classes. Furthermore, L2,1-norm selects the ELM features by disabling some hidden neurons at all time. Section 4.3.8 compares the clustering performance of UFS-ELM with different norms of output weights, and demonstrates the robustness of the L2,1-norm. Chapter 4 investigates an end-to-end clustering algorithm based on extreme learning machine. Empirically, it is proven that the extreme learning machine constrained by manifold regularization can directly approximate the cluster indicators.

Chapter 4 investigates multiview fusion in distance level. The problem is brought out as a pairwise distance has different relative magnitudes among different local structure. The proposed within-view distance fusion produces a more comprehensive distance measure that takes the local geometry structure into account. At the same time, the produced distance measure has a unified scale. Therefore, the multiview distances can be equally fused in nature. Based on the distance fusions, Section 5.2.2 proposes a multiview clustering algorithms adaptively learns

a common graph and derives partition results with desired number of clusters. The proposed DFAN surpasses the state-of-the-art algorithms in multiview clustering, especially the MLAN that shares similar framework. It demonstrates the effectiveness of the distance fusions.

Chapter 5 investigates extreme learning machines as embedding learning method for multiview clustering. Compare the existing multiview spectral clustering methods that constructs graph by measuring the pairwise distances in the original data space. Section 6.2.2 adopts the unsupervised extreme learning machine to learn discriminative data embeddings and fuses the embedding distance with the original distance with equal weights. The fused dual distance significantly boosts the multiview clustering performance on various multiview datasets. Section 6.3.7 empirically shows that the equally combined dual distance results in more reliable clustering performance than unbalance combination of the two distances.

7.2 Future Works

This thesis investigates ELM for spectral clustering and making a step forward to multiview clustering. Some exciting research areas can be further investigated.

This thesis demonstrates that UFS-ELM draws significant performance in clustering. Therefore, it is interesting to extend UFS-ELM to multiview clustering problems. The multiview fusion for UFS-ELM can be done in several ways. First, the graph Laplacian in UFS-ELM is predefined. For the multiview version of UFS-ELM, the graph can be learned from the pairwise similarities measured by the distance fusions. The graph learning can also be done in an auto-weighted and adaptive style, as same as DAMC. The cluster indicator is another multiview information can be fused. However, the cluster indicators are single-class labels that are hard to fuse into a consistent result. Hence, joining the output of the extreme learning machines for multiview clustering is also interesting. As discussed in Chapter 3, UFS-ELM is more computationally complexity than existing ELM methods due to the eigendecomposition. Thus, deploying UFS-ELM in real-time applications is a challenging task. Instead of stopping until a predefined number of iteration, the optimization process of UFS-ELM can be shorten by detecting the convergence of the clustering results.

In both DFAN and DAMC, a trade-off parameter is adaptively tuned to adjust the weights of the soft constraint to ensure the number of components in the graph. However, the parameter is either doubled or halved to satisfy the Laplacian rank constraint. It is obvious that all the values in a specific range can constraint the rank of Laplacian. Therefore, it remains unclear the difference between those values and finding the optimal value for better clustering performance.

With respect to OCA-ELM for 3D shape classification, there are several aspects can be improved. OCA-ELM learns the dataset as a whole, the learned parameters cannot be further tuned based on new data samples. However, in real-time applications, the training samples might be collected one-by-one and the data distribution changes with time. To handle such applications, the classification algorithms must be able to optimizes the parameter based on existing value of parameters and the new data samples (i.e., streaming training or online learning). OCA-ELM can be extended to online learning classification based on online version of ELM, such as online sequence extreme learning machine (OS-ELM) [154] and online sequential fuzzy extreme learning machine (OS-Fuzzy-ELM) [155]. Furthermore, the information of different poses of 3D shape is concatenated in OCA-ELM, pooling or other operations can be utilized to aggregate the poses.

List of Author's Publications

Journal Articles

- **Jichao Chen**, Yijie Zeng, Yue Li, and Guang-Bin Huang. "Unsupervised feature selection based extreme learning machine for clustering." *Neurocomputing* 386 (2020): 198-207.
- **Jichao Chen**, Yijie Zeng, and Guang-Bin Huang. "Distance fusion for multiview clustering." *Neurocomputing* (2021). Under review.
- **Jichao Chen** and Guang-Bin Huang. "Dual distance adaptive multiview clustering." *Neurocomputing* 441 (2021): 311-322.
- Yijie Zeng, **Jchao Chen**, and Guang-Bin Huang. "Slice-Based Online Convolutional Dictionary Learning." *IEEE Transactions on Cybernetics* (2019).
- Yijie Zeng, Yue Li, **Jichao Chen**, Xiaofan Jia, and Guang-Bin Huang. "ELM embedded discriminative dictionary learning for image classification." *Neural Networks* 123 (2020): 331-342.
- Yijie Zeng, **Jichao Chen**, Yue Li, Yuanyuan Qing, and Guang-Bin Huang. "Clustering via Adaptive and Locality-constrained Graph Learning and Unsupervised ELM." *Neurocomputing* (2020).

Conference Proceedings

- **Jichao Chen**, Yijie Zeng, Siqu Wang, Soh Ling Min, and Guang-Bin Huang. "Octree-based Convolutional Autoencoder Extreme Learning Machine for 3D Shape Classification." In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7. IEEE, 2018.
- Yijie Zeng, **Jichao Chen**, and Guang-Bin Huang. "Data Driven Convolutional Sparse Coding for Visual Recognition." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2736-2740. IEEE, 2018.

Bibliography

- [1] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. [7](#)
- [2] Noah A Rosenberg, Terry Burke, Kari Elo, Marcus W Feldman, Paul J Freidlin, Martien AM Groenen, Jossi Hillel, Asko Mäki-Tanila, Michèle Tixier-Boichard, Alain Vignal, et al. Empirical evaluation of genetic clustering methods using multilocus genotypes from 20 chicken breeds. *Genetics*, 159(2):699–713, 2001. [7](#)
- [3] MA Anfyorov. Genetic clustering algorithm. *Russian Technological Journal*, 7(6):134–150, 2020. [7](#)
- [4] George Arimond and Abdulaziz Elfessi. A clustering method for categorical data in tourism market segmentation research. *Journal of Travel Research*, 39(4):391–397, 2001. [7](#)
- [5] Deepali Kamthania, Ashish Pawa, and Srijit S Madhavan. Market segmentation analysis and visualization using k-mode clustering algorithm for e-commerce business. *Journal of computing and information technology*, 26(1): 57–68, 2018. [7](#)
- [6] Guy Barrett Coleman and Harry C Andrews. Image segmentation by clustering. *Proceedings of the IEEE*, 67(5):773–785, 1979. [7](#)
- [7] Thuy Xuan Pham, Patrick Siarry, and Hamouche Oulhadj. Integrating fuzzy entropy clustering with an improved pso for mri brain image segmentation. *Applied Soft Computing*, 65:230–242, 2018. [7](#)
- [8] Chris Ding, Xiaofeng He, Hongyuan Zha, and Horst D Simon. Adaptive dimension reduction for clustering high dimensional data. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 147–154. IEEE, 2002. [7](#)
- [9] Amir Karami. Application of fuzzy clustering for text data dimensionality reduction. *International Journal of Knowledge Engineering and Data Mining*, 6(3):289–306, 2019. [7](#)

- [10] Fan Hu, Gui-Song Xia, Zifeng Wang, Xin Huang, Liangpei Zhang, and Hong Sun. Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(5), 2015. [7](#)
- [11] Wei Yao, Cheng Lian, and Lorenzo Bruzzone. Clustercnn: Clustering-based feature learning for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 2020. [7](#)
- [12] Steven K Thompson. Adaptive cluster sampling. *Journal of the American Statistical Association*, 85(412):1050–1059, 1990. [7](#)
- [13] Xiaoxue Feng, Xiaohui Liang, and Zili Zhang. A cluster sampling method for image matting via sparse coding. In *European Conference on Computer Vision*, pages 204–219. Springer, 2016. [7](#)
- [14] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. [7](#)
- [15] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1, pages 291–324, 2002. [7](#)
- [16] Mehrbakhsh Nilashi, Karamollah Bagherifard, Mohsen Rahmani, and Vahid Rafe. A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques. *Computers & industrial engineering*, 109:357–368, 2017. [7](#)
- [17] L Douglas Bakeryz and Andrew Kachites McCallumyz. Distributional clustering of words for text classification. In *Proceedings of SIGIR*. Citeseer, 1998. [7](#)
- [18] Inderjit S Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3(Mar):1265–1287, 2003.
- [19] Niusha Shafiabady, Lam Hong Lee, Rajprasad Rajkumar, VP Kallimani, Nik Ahmad Akram, and Dino Isa. Using unsupervised clustering approach to train the support vector machine for text classification. *Neurocomputing*, 211:4–10, 2016. [7](#)
- [20] Feiping Nie, Xiaoqian Wang, Michael I Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. [9](#), [66](#)
- [21] Bojan Mohar, Y Alavi, G Chartrand, and OR Oellermann. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991. [9](#)

- [22] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. [9](#), [97](#)
- [23] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010. [10](#), [38](#), [40](#), [41](#)
- [24] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2006.
- [25] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1151–1157. ACM, 2007.
- [26] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992. [10](#)
- [27] Jennifer G Dy and Carla E Brodley. Feature selection for unsupervised learning. *Journal of machine learning research*, 5(Aug):845–889, 2004. [10](#)
- [28] Mihaela Breaban and Henri Luchian. A unifying criterion for unsupervised clustering and feature selection. *Pattern Recognition*, 44(4):854–865, 2011. [10](#)
- [29] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. L2, 1-norm regularized discriminative feature selection for unsupervised. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011. [10](#), [11](#), [38](#)
- [30] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. Unsupervised feature selection using nonnegative spectral analysis. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. [10](#), [11](#), [38](#)
- [31] Mingjie Qian and Chengxiang Zhai. Robust unsupervised feature selection. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013. [10](#), [11](#), [38](#)
- [32] Suhang Wang, Jiliang Tang, and Huan Liu. Embedded unsupervised feature selection. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015. [10](#)
- [33] Lei Zhang and David Zhang. Robust visual knowledge transfer via extreme learning machine-based domain adaptation. *IEEE Transactions on Image Processing*, 25(10):4959–4973, 2016. [10](#), [38](#)
- [34] Xiong Luo, Xiaohui Chang, and Xiaojuan Ban. Regression and classification using extreme learning machine based on l1-norm and l2-norm. *Neurocomputing*, 174:179–186, 2016.

- [35] Sihang Zhou, Xinwang Liu, Qiang Liu, Siqi Wang, Chengzhang Zhu, and Jianping Yin. Random fourier extreme learning machine with l2, 1-norm regularization. *Neurocomputing*, 174:143–153, 2016.
- [36] Tan Guo, Lei Zhang, and Xiaoheng Tan. Neuron pruning-based discriminative extreme learning machine for pattern classification. *Cognitive Computation*, 9(4):581–595, 2017. [10](#), [38](#)
- [37] Zahra Ezzati Khatab, Amirhosein Hajihoseini, and Seyed Ali Ghorashi. A fingerprint method for indoor localization using autoencoder based deep extreme learning machine. *IEEE sensors letters*, 2(1):1–4, 2017. [10](#), [38](#)
- [38] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. [11](#)
- [39] Yijie Zeng, Yue Li, Jichao Chen, Xiaofan Jia, and Guang-Bin Huang. Elm embedded discriminative dictionary learning for image classification. *Neural Networks*, 123:331–342, 2020. [12](#)
- [40] Jichao Chen, Yijie Zeng, Siqi Wang, Soh Ling Min, and Guang-Bin Huang. Octree-based convolutional autoencoder extreme learning machine for 3d shape classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2018. [12](#), [16](#)
- [41] Siqi Wang, En Zhu, Xiping Hu, Xinwang Liu, Qiang Liu, Jianping Yin, and Fei Wang. Robustness can be cheap: A highly efficient approach to discover outliers under high outlier ratios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5313–5320, 2019. [12](#), [16](#)
- [42] Yijie Zeng, Jichao Chen, Yue Li, Yuanyuan Qing, and Guang-Bin Huang. Clustering via adaptive and locality-constrained graph learning and unsupervised elm. *Neurocomputing*, 2020. [16](#)
- [43] Jichao Chen, Yijie Zeng, Yue Li, and Guang-Bin Huang. Unsupervised feature selection based extreme learning machine for clustering. *Neurocomputing*, 386:198–207, 2020. [16](#)
- [44] Siqi Wang, Qiang Liu, En Zhu, Jianping Yin, and Wentao Zhao. Mst-gen: An efficient parameter selection method for one-class extreme learning machine. *IEEE transactions on cybernetics*, 47(10):3266–3279, 2017. [12](#), [16](#)
- [45] Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Guang-Bin Huang, and Chi Man Vong. Representational learning with extreme learning machine for big data. *IEEE intelligent systems*, 28(6):31–34, 2013. [12](#), [25](#)
- [46] Dongshun Cui, Guanghao Zhang, Wei Han, Liyanaarachchi Lekamalage Chamara Kasun, Kai Hu, and Guang-Bin Huang. Compact feature representation for image classification using elms. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1015–1022, 2017.

- [47] Dongshun Cui, Guang-Bin Huang, LL Chamara Kasun, Guanghao Zhang, and Wei Han. Elmnet: feature learning using extreme learning machines. In *2017 IEEE international conference on image processing (ICIP)*, pages 1857–1861. IEEE, 2017.
- [48] Guanghao Zhang, Dongshun Cui, Shangbo Mao, and Guang-Bin Huang. Sparse bayesian learning for extreme learning machine auto-encoder. In *International Conference on Extreme Learning Machine*, pages 319–327. Springer, 2018. [12](#)
- [49] Liyanaarachchi Lekamalage Chamara Kasun, Yan Yang, Guang-Bin Huang, and Zhengyou Zhang. Dimension reduction with extreme learning machine. *IEEE transactions on Image Processing*, 25(8):3906–3918, 2016. [12](#)
- [50] Yueqing Wang, Zhige Xie, Kai Xu, Yong Dou, and Yuanwu Lei. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. *Neurocomputing*, 174:988–998, 2016. [12](#), [24](#), [33](#)
- [51] Zhiyong Huang, Yuanlong Yu, Jason Gu, and Huaping Liu. An efficient method for traffic sign recognition based on extreme learning machine. *IEEE transactions on cybernetics*, 47(4):920–933, 2016. [12](#)
- [52] Yu Zhang, Yu Wang, Guoxu Zhou, Jing Jin, Bei Wang, Xingyu Wang, and Andrzej Cichocki. Multi-kernel extreme learning machine for eeg classification in brain-computer interfaces. *Expert Systems with Applications*, 96: 302–310, 2018. [12](#)
- [53] WK Wong and ZX Guo. A hybrid intelligent model for medium-term sales forecasting in fashion retail supply chains using extreme learning machine and harmony search algorithm. *International Journal of Production Economics*, 128(2):614–624, 2010. [12](#)
- [54] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011. [12](#), [40](#), [120](#)
- [55] Guang-Bin Huang, Qin-Yu Zhu, Chee-Kheong Siew, et al. Extreme learning machine: a new learning scheme of feedforward neural networks. *Neural networks*, 2:985–990, 2004. [13](#), [27](#), [38](#)
- [56] KS Banerjee. Generalized inverse of matrices and its applications, 1973. [13](#)
- [57] W. Deng, Q. Zheng, and L. Chen. Regularized Extreme Learning Machine. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 389–395, March 2009. [13](#), [38](#)
- [58] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse. Op-elm: optimally pruned extreme learning machine. *IEEE transactions on neural networks*, 21(1):158–162, 2009. [13](#), [38](#)

- [59] Timo Similä and Jarkko Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *International Conference on Artificial Neural Networks*, pages 97–102. Springer, 2005. [13](#), [38](#)
- [60] Yoan Miche, Mark Van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse. Trop-elm: a double-regularized elm using lars and tikhonov regularization. *Neurocomputing*, 74(16):2413–2421, 2011. [13](#), [38](#)
- [61] David M Allen. The relationship between variable selection and data augmentation and a method for prediction. *technometrics*, 16(1):125–127, 1974. [13](#)
- [62] Guang-Bin Huang, Lei Chen, Chee Kheong Siew, et al. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17(4):879–892, 2006. [14](#)
- [63] Guang-Bin Huang and Lei Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16-18):3056–3062, 2007. [14](#)
- [64] Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. Extreme learning machine for joint embedding and clustering. *Neurocomputing*, 277:78–88, 2018. [14](#), [39](#), [46](#), [92](#)
- [65] Gao Huang, Shiji Song, Jatinder ND Gupta, and Cheng Wu. Semi-supervised and unsupervised extreme learning machines. *IEEE transactions on cybernetics*, 44(12):2405–2417, 2014. [14](#), [46](#)
- [66] Gao Huang, Tianchi Liu, Yan Yang, Zhiping Lin, Shiji Song, and Cheng Wu. Discriminative clustering via extreme learning machine. *Neural Networks*, 70:1–8, 2015. [14](#), [46](#)
- [67] Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. An adaptive graph learning method based on dual data representations for clustering. *Pattern Recognition*, 77:126–139, 2018. [15](#), [92](#), [94](#)
- [68] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. [16](#)
- [69] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, volume 1, pages 886–893. IEEE, 2005. [16](#)
- [70] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996. [16](#)

- [71] Zhige Xie, Kai Xu, Wen Shan, Ligang Liu, Yueshan Xiong, and Hui Huang. Projective feature learning for 3d shapes with multi-view depth images. In *Computer Graphics Forum*, volume 34, pages 1–11. Wiley Online Library, 2015. [16](#), [22](#), [33](#)
- [72] Kun Zhan, Feiping Nie, Jing Wang, and Yi Yang. Multiview consensus graph clustering. *IEEE Transactions on Image Processing*, 28(3):1261–1270, 2018. [16](#), [71](#), [73](#), [105](#)
- [73] Muli Yang, Cheng Deng, and Feiping Nie. Adaptive-weighting discriminative regression for multi-view classification. *Pattern Recognition*, 88:236–245, 2019. [16](#)
- [74] Sham M Kakade and Dean P Foster. Multi-view regression via canonical correlation analysis. In *International Conference on Computational Learning Theory*, pages 82–96. Springer, 2007. [16](#)
- [75] Grigorios Tzortzis and Aristidis Likas. Kernel-based weighted multi-view clustering. In *2012 IEEE 12th international conference on data mining*, pages 675–684. IEEE, 2012. [17](#)
- [76] Dongyan Guo, Jian Zhang, Xinwang Liu, Ying Cui, and Chunxia Zhao. Multiple kernel learning based multi-view spectral clustering. In *2014 22nd International Conference on Pattern Recognition*, pages 3774–3779. IEEE, 2014. [17](#)
- [77] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. Multiple kernel k-means clustering with matrix-induced regularization. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. [17](#)
- [78] Xiaochun Cao, Changqing Zhang, Huazhu Fu, Si Liu, and Hua Zhang. Diversity-induced multi-view subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–594, 2015. [17](#)
- [79] Xuran Zhao, Nicholas Evans, and Jean-Luc Dugelay. A subspace co-training framework for multi-view clustering. *Pattern Recognition Letters*, 41:73–82, 2014.
- [80] Qiang Wang, Yong Dou, Xinwang Liu, Qi Lv, and Shijie Li. Multi-view clustering with extreme learning machine. *Neurocomputing*, 214:483–494, 2016.
- [81] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2): 52–68, 2011. [17](#)
- [82] Yang Wang, Xuemin Lin, Lin Wu, Wenjie Zhang, Qing Zhang, and Xiaodi Huang. Robust subspace clustering for multi-view data by exploiting correlation consensus. *IEEE Transactions on Image Processing*, 24(11):3939–3949, 2015.

- [83] Zhao Kang, Wangtao Zhou, Zhitong Zhao, Junming Shao, Meng Han, and Zenglin Xu. Large-scale multi-view subspace clustering in linear time. In *AAAI*, pages 4412–4419, 2020. [17](#)
- [84] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260. SIAM, 2013. [17](#)
- [85] Xiao Cai, Feiping Nie, and Heng Huang. Multi-view k-means clustering on big data. In *Twenty-Third International Joint conference on artificial intelligence*, 2013.
- [86] Weihua Ou, Shujian Yu, Gai Li, Jian Lu, Kesheng Zhang, and Gang Xie. Multi-view non-negative matrix factorization by patch alignment framework with view consistency. *Neurocomputing*, 204:116–124, 2016.
- [87] Handong Zhao, Zhengming Ding, and Yun Fu. Multi-view clustering via deep matrix factorization. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. [17](#)
- [88] Feiping Nie, Jing Li, Xuelong Li, et al. Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification. In *IJCAI*, pages 1881–1887, 2016. [17](#), [19](#)
- [89] Feiping Nie, Jing Li, Xuelong Li, et al. Self-weighted multiview clustering with multiple graphs. In *IJCAI*, pages 2564–2570, 2017. [17](#), [19](#), [73](#)
- [90] Kun Zhan, Changqing Zhang, Junpeng Guan, and Junsheng Wang. Graph learning for multiview clustering. *IEEE transactions on cybernetics*, 48(10): 2887–2895, 2017. [17](#), [19](#)
- [91] Zhao Kang, Haiqi Pan, Steven CH Hoi, and Zenglin Xu. Robust graph learning from noisy data. *IEEE transactions on cybernetics*, 50(5):1833–1843, 2019. [17](#)
- [92] Grigorios F Tzortzis and Aristidis C Likas. The global kernel k -means algorithm for clustering in feature space. *IEEE transactions on neural networks*, 20(7):1181–1194, 2009. [17](#)
- [93] Bin Shen and Luo Si. Non-negative matrix factorization clustering on multiple manifolds. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. [17](#)
- [94] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 4238–4246, 2015. [17](#)
- [95] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002. [17](#)

- [96] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005. [17](#)
- [97] Hao Wang, Yan Yang, and Bing Liu. Gmc: Graph-based multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1116–1129, 2019. [17](#)
- [98] Zhao Kang, Guoxin Shi, Shudong Huang, Wenyu Chen, Xiaorong Pu, Joey Tianyi Zhou, and Zenglin Xu. Multi-graph fusion for multi-view spectral clustering. *Knowledge-Based Systems*, 189:105102, 2020. [17](#)
- [99] Xi Peng, Zhenyu Huang, Jiancheng Lv, Hongyuan Zhu, and Joey Tianyi Zhou. Comic: Multi-view clustering without parameter selection. In *International Conference on Machine Learning*, pages 5092–5101, 2019. [18](#)
- [100] Yijie Lin, Yuanbiao Gou, Zitao Liu, Boyun Li, Jiancheng Lv, and Xi Peng. Completer: Incomplete multi-view clustering via contrastive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11174–11183, 2021. [18](#)
- [101] Jie Wen, Zheng Zhang, Zhao Zhang, Lunke Fei, and Meng Wang. Generalized incomplete multiview clustering with flexible locality structure diffusion. *IEEE transactions on cybernetics*, 51(1):101–114, 2020. [18](#)
- [102] Xiaobo Zhang, Yan Yang, Tianrui Li, Yiling Zhang, Hao Wang, and Hamido Fujita. Cmc: A consensus multi-view clustering model for predicting alzheimer’s disease progression. *Computer Methods and Programs in Biomedicine*, 199:105895, 2021. [18](#)
- [103] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, pages 1413–1421, 2011. [19](#), [72](#), [104](#)
- [104] Feiping Nie, Guohao Cai, and Xuelong Li. Multi-view clustering and semi-supervised classification with adaptive neighbours. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. [19](#), [20](#), [72](#), [92](#), [105](#)
- [105] Kun Zhan, Chaoxi Niu, Changlu Chen, Feiping Nie, Changqing Zhang, and Yi Yang. Graph structure fusion for multiview clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2018. [19](#), [60](#), [73](#), [104](#)
- [106] Wayne Carlson. A critical history of computer graphics and animation. *The Ohio State University*, 2003. [21](#)
- [107] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. Sph fluids in computer graphics. 2014. [21](#)
- [108] Francis S Hill Jr. *Computer graphics using open gl*. Pearson Education, 2008. [21](#)

- [109] Hazel Gardiner and Charlie Gere. *Art practice in a digital culture*. Routledge, 2016. [21](#)
- [110] M Lo Brutto and Paola Meli. Computer vision tools for 3d modelling in archaeology. *International Journal of Heritage in the Digital Era*, 1(1_suppl): 1–6, 2012. [21](#)
- [111] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. [21](#)
- [112] Rui Huang, Shusheng Zhang, Xiaoliang Bai, Changhong Xu, and Bo Huang. An effective subpart retrieval approach of 3d cad models for manufacturing process reuse. *Computers in industry*, 67:38–53, 2015. [21](#)
- [113] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. [21](#)
- [114] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. [21](#)
- [115] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. [21](#)
- [116] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018. [21](#)
- [117] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5023–5032, 2016. [21](#)
- [118] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015. [21](#), [33](#)
- [119] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003. [21](#), [33](#)

- [120] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016. [22](#), [33](#)
- [121] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020. [22](#)
- [122] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. [22](#)
- [123] Sudhakar Kumawat and Shanmuganathan Raman. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4903–4912, 2019. [22](#)
- [124] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Fine-grained 3d shape classification with hierarchical part-view attention. *IEEE Transactions on Image Processing*, 30:1744–1758, 2021. [22](#)
- [125] Yong Xu, Chaoda Zheng, Ruotao Xua, Yuhui Quan, and Haibin Ling. Multi-view 3d shape recognition via correspondence-aware deep learning. *IEEE Transactions on Image Processing*, 2021. [22](#)
- [126] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. [24](#), [31](#), [33](#)
- [127] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. [24](#), [33](#), [34](#)
- [128] Alberto Garcia-Garcia, Francisco Gomez-Donoso, Jose Garcia-Rodriguez, Sergio Orts-Escolano, Miguel Cazorla, and J Azorin-Lopez. Pointnet: A 3d convolutional neural network for real-time object class recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1584. IEEE, 2016. [24](#)
- [129] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. [24](#), [33](#)

- [130] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003. [24](#), [33](#)
- [131] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017. [24](#), [31](#), [33](#), [34](#)
- [132] Guang-Bin Huang. What are extreme learning machines? filling the gap between frank rosenblatt’s dream and john von neumann’s puzzle. *Cognitive Computation*, 7(3):263–278, 2015. [25](#)
- [133] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016. [33](#)
- [134] Chenping Hou, Feiping Nie, Dongyun Yi, and Dacheng Tao. Discriminative embedded clustering: A framework for grouping high-dimensional data. *IEEE transactions on neural networks and learning systems*, 26(6):1287–1299, 2014. [39](#), [92](#)
- [135] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006. [40](#), [92](#)
- [136] Daniel B. Graham and Nigel M. Allinson. Characterising Virtual Eigensignatures for General Purpose Face Recognition. *Face Recognition*, pages 446–456, 1998. doi: 10.1007/978-3-642-72201-1_25. [44](#)
- [137] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression (pie) database. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 53–58. IEEE, 2002. [44](#)
- [138] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001. [44](#)
- [139] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the world wide web. *Artificial intelligence*, 118(1-2):69–113, 2000. [44](#)
- [140] Ananda L Freire, Guilherme A Barreto, Marcus Veloso, and Antonio T Varela. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *2009 6th Latin American Robotics Symposium (LARS 2009)*, pages 1–6. IEEE, 2009. [44](#)

- [141] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. [52](#)
- [142] Ky Fan. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences of the United States of America*, 35(11):652, 1949. [64](#), [99](#)
- [143] James C Bezdek and Richard J Hathaway. Convergence of alternating optimization. *Neural, Parallel & Scientific Computations*, 11(4):351–368, 2003. [65](#)
- [144] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. [66](#)
- [145] Martijn van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998. [71](#), [104](#)
- [146] ALIREZA Khotanzad and Yaw Hua Hong. Rotation invariant pattern recognition using zernike moments. In *[1988 Proceedings] 9th International Conference on Pattern Recognition*, pages 326–328. IEEE, 1988. [71](#)
- [147] SA Nene, SK Nayar, and H Murase. Columbia object image library: Coil-20. *Dept. Comp. Sci., Columbia University, New York, Tech. Rep*, 1996. [71](#), [104](#)
- [148] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML’06)*, pages 377–384. ACM Press, 2006. [72](#)
- [149] Xiao Cai, Feiping Nie, Heng Huang, and Farhad Kamangar. Heterogeneous image feature integration via multi-modal spectral clustering. In *CVPR 2011*, pages 1977–1984. IEEE, 2011. [72](#), [104](#)
- [150] Zhao Kang, Xinjia Zhao, Chong Peng, Hongyuan Zhu, Joey Tianyi Zhou, Xi Peng, Wenyu Chen, and Zenglin Xu. Partition level multiview subspace clustering. *Neural Networks*, 122:279–288, 2020. [73](#)
- [151] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. [74](#), [105](#)
- [152] Sergios Theodoridis and Konstantinos Koutroumbas. Pattern recognition. 2003. *Elsevier Inc*, 2009. [92](#)
- [153] Rocco Langone, Carlos Alzate, Bart De Ketelaere, Jonas Vlasselaer, Wannes Meert, and Johan AK Suykens. Ls-svm based spectral clustering and regression for predicting maintenance of industrial machines. *Engineering Applications of Artificial Intelligence*, 37:268–278, 2015. [92](#)

- [154] Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006. [122](#)
- [155] Hai-Jun Rong, Guang-Bin Huang, Narasimhan Sundararajan, and Paramasivan Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, 2009. [122](#)