

eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

**Contribution to Graph-based Multi-view Clustering:
Algorithms and Applications**

Author: Sally El Hajjar

Supervisor

Fadi Dornaika

12/10/2022

ACKNOWLEDGEMENT

First, I would like to thank my thesis director, Dr. Fadi Dornaika, for his availability, his patience in guiding me, and above all, his judicious advice, which contributed to feeding my ideas in the field of clustering. All the exchanges that I had with my director allowed me to progress quickly in my research. In addition, I would like to thank Dr. Fahed Abdallah, who was always ready to help me throughout my thesis and to advise me when I needed help.

I also thank my parents, and my friends for their continued encouragement and for their support, which has made me what I am today. In addition, I would like to thank UPV/EHU University for giving me the opportunity to continue my doctoral studies at this university.

To all these speakers (and others too many to mention), I offer my thanks, respect, and gratitude.

ABSTRACT

In this thesis, we study unsupervised learning, specifically, clustering methods for dividing data into meaningful groups. A major challenge is to find an efficient algorithm with low computational complexity that can handle different types and sizes of data sets.

For this purpose, we propose two approaches. The first approach is named "Multi-view Clustering via Kernelized Graph and Nonnegative Embedding" (MKGNE), and the second approach is called "Multi-view Clustering via Consensus Graph Learning and Nonnegative Embedding" (MVCGE). These two approaches jointly solve four tasks. They jointly estimate the unified similarity matrix over all views using the kernel tricks, the unified spectral projection of the data, the cluster indicator matrix, and the weight of each view without additional parameters. With these two approaches, there is no need for any postprocessing such as k-means clustering.

In a further study, we propose a method named "Multi-view Spectral Clustering via Constrained Nonnegative Embedding" (CNESE). This method can overcome the drawbacks of the spectral clustering approaches, since they only provide a nonlinear projection of the data, on which an additional step of clustering is required. This can degrade the quality of the final clustering due to various factors such as the initialization process or outliers. Overcoming these drawbacks can be done by introducing a nonnegative embedding matrix which gives the final clustering assignment. In addition, some constraints are added to the targeted matrix to enhance the clustering performance.

In accordance with the above methods, a new method called "Multi-view Spectral Clustering with a self-taught Robust Graph Learning" (MCSRGL) has been developed. Different from other approaches, this method integrates two main paradigms into the one-step multi-view clustering model. First, we construct an additional graph by using the cluster label space in addition to the graphs associated with the data space. Second, a smoothness constraint is exploited to constrain the cluster-label matrix and make it more consistent with the data views and the label view.

Moreover, we propose two unified frameworks for multi-view clustering in Chapter 9. In these framework, we attempt to determine a view based graphs, the consensus graph, the consensus spectral representation, and the soft clustering assignments. These methods retain the main advantages of the aforementioned methods and integrate the concepts of consensus and unified matrices. By using the unified matrices, we enforce the matrices of different views to be similar, and thus the problem of noise and inconsistency between different views will be reduced.

Extensive experiments were conducted on several public datasets with different types and sizes, varying from face image datasets, to document datasets, handwritten datasets, and synthetics datasets. We provide several analyses of the proposed algorithms, including ablation studies, hyper-parameter sensitivity analyses, and computational costs. The experimental results show that the developed algorithms through this thesis are relevant and outperform several competing methods.

Keywords: Machine learning, unsupervised learning, multi-view clustering, graph learning, spectral projection, nonnegative embedding, auto-weighted strategy, clustering algorithms, similarity graph, graph construction, soft cluster assignments, cluster label space, consensus matrices, constrained nonnegative embedding, smoothness constraints.

RESUMEN

En esta tesis, estudiamos el aprendizaje no supervisado, específicamente, los métodos de agrupamiento para dividir datos en grupos significativos. Un desafío importante es cómo encontrar un algoritmo eficiente con baja complejidad computacional para manejar diferentes tipos y tamaños de conjuntos de datos.

Para ello, proponemos dos enfoques. El primer enfoque se denomina "Agrupación de múltiples vistas a través de gráficos kernelizados e incrustaciones no negativas" (MKGNE), y el segundo enfoque se denomina "Agrupación de múltiples vistas a través del aprendizaje de gráficos de consenso e incrustaciones no negativas" (MVCGE). Estos dos enfoques resuelven conjuntamente cuatro tareas. Estiman conjuntamente la matriz de similitud unificada sobre todas las vistas utilizando los trucos del núcleo, la proyección espectral unificada de los datos, la matriz de indicadores de grupo y el peso de cada vista sin parámetros adicionales. Con estos dos enfoques, no hay necesidad de ningún procesamiento posterior, como el agrupamiento de k-means.

En un estudio adicional, proponemos un método denominado "Agrupación espectral de vista múltiple a través de incrustaciones no negativas restringidas" (CNESE). Este método puede superar los inconvenientes de los enfoques de agrupamiento espectral, ya que solo proporcionan una proyección no lineal de los datos, en la que se requiere un paso adicional de agrupamiento. Esto puede degradar la calidad del agrupamiento final debido a varios factores, como el proceso de inicialización o los valores atípicos. Se pueden superar estos inconvenientes mediante la introducción de una matriz de incrustación no negativa que proporcione la asignación de agrupación final. Además, se agregan algunas restricciones a la matriz objetivo para mejorar el rendimiento de la agrupación.

De acuerdo con los métodos anteriores, se ha desarrollado un nuevo método denominado "Agrupación espectral multivista con un aprendizaje gráfico robusto autodidacta" (MCSRGL). A diferencia de otros enfoques, este método integra dos paradigmas principales en el modelo de agrupamiento de vistas

múltiples de un solo paso. Primero, construimos un gráfico adicional utilizando el espacio de etiquetas de clúster además de los gráficos asociados con el espacio de datos. En segundo lugar, se explota una restricción de suavidad para restringir la matriz de etiquetas de clúster y hacerla más coherente con las vistas de datos y la vista de etiquetas.

Además, proponemos dos marcos unificados para el agrupamiento de vistas múltiples en el Capítulo 9. En este marco, intentamos determinar gráficos basados en vistas, el gráfico de consenso, la representación espectral de consenso y las asignaciones de agrupamiento suave. Estos métodos conservan las principales ventajas de los métodos antes mencionados e integran los conceptos de consenso y matrices unificadas. Al usar las matrices unificadas, hacemos que las matrices de diferentes vistas sean similares y, por lo tanto, se reducirá el problema del ruido y la inconsistencia entre diferentes vistas.

Además, muchos otros trabajos y proyectos relacionados con métodos de aprendizaje automático ya se han completado y se mencionan en esta tesis.

Se realizaron extensos experimentos en varios conjuntos de datos públicos con diferentes tipos y tamaños, que variaban desde conjuntos de datos de imágenes faciales hasta conjuntos de datos de documentos, conjuntos de datos escritos a mano y conjuntos de datos sintéticos. Proporcionamos varios análisis de los algoritmos propuestos, incluidos estudios de ablación, análisis de sensibilidad de hiperparámetros y costos computacionales. Los resultados experimentales muestran que los algoritmos desarrollados a través de esta tesis son relevantes y superan a varios métodos de la competencia.

Palabras clave: Aprendizaje automático, aprendizaje no supervisado, agrupamiento de vistas múltiples, aprendizaje de gráficos, proyección espectral, incrustación no negativa, estrategia ponderada automáticamente, algoritmos de agrupamiento, gráfico de similitud, construcción de gráficos, asignaciones de clúster flexibles, espacio de etiquetas de clúster, matrices de consenso, incrustación no negativa restringida, restricciones de suavidad.

Contents

Contents	vii
List of Figures	xi
List of Tables	xiv
Notations	xvii
1 General Introduction to multi-view learning	1
1.1 General Introduction	1
1.2 Benefits of multi-view Learning	3
1.2.1 Practical problems of multi-view learning approaches	5
1.3 Research outline and manuscript structure	5
2 General Introduction to machine learning concepts	9
2.1 Different types of machine learning methods	9
2.2 Overview of Kernel methods	19
2.3 Overview of Matrix Factorization	23
2.3.1 Background	23
2.3.2 Clustering property of the NMF method	24
2.4 Overview of the Singular Value Decomposition algorithm	24
2.5 Overview of the t-SNE method	25
2.6 Overview of the Gradient Descent method	25
2.6.1 How Gradient Descent works	28
2.7 Overview of the Auto-weighted strategies	29
2.8 Overview of the Cluster Evaluation Metrics	30

2.9	Overview of Deep Learning	34
2.9.1	Deep Learning work	35
2.9.2	The main difference between Deep Learning and Machine Learning	36
2.9.3	Convolutional neural networks	36
2.9.4	Definition of some descriptors	39
3	Related work	45
3.1	Related Work	45
3.2	Typical approaches	56
4	Experimental setup and Datasets	59
4.1	Motivation	59
4.2	Datasets	59
4.2.1	Description of the datasets used in our methods	59
4.3	Experimental Setup	64
5	Direct Multi-view Spectral Clustering with Consistent Kernelized Graph and Con- volved Nonnegative Representation	67
5.1	Proposed Approach	68
5.1.1	Optimization	71
5.1.2	Differences with state-of-the-art methods	74
5.1.3	Convergence analysis	74
5.2	Performance Evaluation	75
5.2.1	Datasets	75
5.2.2	Experimental Setup	75
5.2.3	Experimental results	77
5.2.4	Parameter sensitivity	77
5.2.5	Analysis of the results and method comparison	79
5.2.6	Convergence study	82
5.2.7	Sequential estimation vs. proposed approach	83
5.2.8	Computational complexity	83
5.2.9	Clustering visualization	84
5.3	Conclusion	85

6	Consensus graph and spectral representation for one-step multi-view kernel based clustering	87
6.1	Proposed Approach	88
6.2	Performance Evaluation	92
6.2.1	Datasets	92
6.2.2	Experimental setup	92
6.2.3	Experimental results	93
6.2.4	Ablation study	93
6.2.5	Analysis of results and method comparison	96
6.2.6	Clustering visualization	97
6.3	Conclusion	97
7	Multi-view Spectral Clustering via Constrained Nonnegative Embedding	101
7.1	Proposed Approach	102
7.1.1	Optimization	103
7.2	Performance Evaluation	105
7.2.1	Experimental Setup	105
7.2.2	Experimental results	106
7.2.3	Parameter sensitivity and ablation study	107
7.2.4	Analysis of results and method comparison	109
7.2.5	Convergence study	110
7.2.6	Computational complexity analysis and time cost	111
7.3	Conclusion	112
8	One-step Multi-view Spectral Clustering with Cluster Label Correlation Graph	113
8.1	Proposed Approach	114
8.1.1	Incorporating cluster label space	116
8.1.2	Optimization	118
8.1.3	Convergence analysis	119
8.2	Performance Evaluation	121
8.2.1	Experimental setup	121
8.2.2	Experimental results	121
8.2.3	Ablation study and parameter sensitivity	121
8.2.4	Analysis of results and method comparison	123

8.2.5	Convergence study	125
8.2.6	Clustering visualization	125
8.3	Conclusion	126
9	A Unified Framework for Multi-view Clustering via Consensus Graph and Spectral Representation Learning	129
9.1	Proposed Approach	130
9.2	Optimization of the model	134
9.2.1	Optimization of OSMGSCA (Eq. (9.7))	134
9.2.2	Optimization of U-MCJGLNMA (Eq. (9.8))	137
9.3	Performance Evaluation	139
9.3.1	Experimental Setup	139
9.3.2	Experimental results	140
9.3.3	Ablation study	141
9.3.4	Parameter sensitivity	142
9.3.5	Analysis of results and method comparison	144
9.3.6	Computational complexity	145
9.3.7	Convergence Study	146
9.4	Clustering visualization	147
9.5	Conclusion	148
10	Conclusion and future work	151
10.1	Conclusion	151
10.2	Perspectives	152
11	Publications	155
11.1	International Journals:	155
11.2	International Conferences:	155
	Bibliography	157

List of Figures

1.1	Disordered data before clustering (left) and ordered after clustering (right) [1].	2
1.2	Reducing the effect of noise.	4
2.1	Different similarity graphs [2].	16
2.2	Undirected graph and its Laplacian matrix.	17
2.3	Feature map of the data set to a higher space.	20
2.4	Illustration of the concept of the SVD method.	26
2.5	Illustration of the MNIST data set after applying t-SNE.	27
2.6	Illustration of the steps taken to reach the mountain from top to bottom.	27
2.7	The iterative procedure of the gradient descent method [3].	29
2.8	The impact of the learning rate parameter [4].	30
2.9	Clustering example.	32
2.10	Difference between AI, ML and DL [5].	35
2.11	An example of the convolution operation.	37
2.12	An example of the Padding and Stride operations.	38
2.13	An example of the Pooling operation.	39
2.14	Simple Convolutional Neural Network architecture [6].	39
2.15	An example of the Local binary patterns feature computation [7].	40
2.16	An example of the HOG feature extraction [8].	41
2.17	Visualization of the concept of the ResNet descriptor [9].	42
2.18	Visualization of the concept of the DenseNet descriptor [9].	43
4.1	Typical images in different datasets.	63
4.2	Lung X-Rays images for the three mentioned classes: (a): Normal, (b) Pneumonia, and (c) Covid-19.	64

4.3	Visualization of the original synthetic datasets: (a) Tetra, (b) Hepta, and (c) Chainlink [10].	64
5.1	Illustration of the MKGNE method.	69
5.2	Clustering performance as a function of the balance parameters using the ORL dataset. (a) and (c) depict ACC (%). (b) and (d) depict NMI (%).	80
5.3	Clustering performance as a function of the balance parameters using the MSRCv1 dataset. (a) and (c) depict ACC (%). (b) and (d) depict NMI (%).	81
5.4	Objective function versus iteration number on four different datasets.	82
5.5	Four clusters provided by our approach for the ORL dataset.	85
5.6	t-SNE of the original features and the spectral projection of ORL dataset.	86
6.1	t-SNE of the spectral projection and nonnegative embedding matrices obtained by the proposed clustering method MVCGE for different datasets.	98
6.2	Visualization of the two clusters obtained by three different methods for the Chainlink dataset.	99
7.1	Clustering performance ACC (%) (left column) and NMI (%) (right column) as a function of α and λ on the COIL20, ORL, MSRCv1, and BBCSport datasets.	108
7.2	Convergence of CNESE on the MSRCv1 dataset.	111
8.1	Illustration of the MCSRGL method.	117
8.2	Clustering performance ACC (%) and NMI (%) as a function of λ on the COIL20 and ORL datasets.	123
8.3	Convergence of MCSRGL on the COIL20, ORL, Out-Scene, and NUS datasets.	125
8.4	t-SNE visualization of the spectral projection, and cluster label matrices on the COIL20, and ORL datasets.	127
9.1	Illustration of the OSMGSCA method.	133
9.2	Illustration of the U-MCJGLNMA method.	134
9.3	Clustering performance ACC (%) and NMI (%) of the OSMGSCA method as a function of λ_1 , λ_2 on the ORL and MSRCv1 datasets.	145
9.4	Clustering performance ACC (%) and NMI (%) of the U-MCJGLNMA method as a function of λ_1 , λ_2 on the ORL and MSRCv1 datasets.	146
9.5	Clustering performance ACC (%) and NMI (%) as a function of λ_3 on the ORL and MSRCv1 datasets: (a): OSMGSCA, (b) U-MCJGLNMA.	147
9.6	Clustering performance ACC (%) and NMI (%) as a function of K on the ORL and MSRCv1 datasets: (a): OSMGSCA, (b): U-MCJGLNMA.	148

9.7 Convergence of our two proposed methods: (a) OSMGSCA and (b) U-MCJGLNMA on the
 ORL and MSRCv1 datasets. 149

9.8 t-SNE visualization of the clustering obtained by four different methods on the ORL dataset. 150

List of Tables

2.1	Example of ground truth and the obtained cluster labels.	34
3.1	Some related multi-view clustering methods.	53
4.1	Description of the datasets.	62
4.2	Description of the datasets.	62
5.1	Clustering performance on the COIL20, ORL, Out-Scene, and NUS datasets.	78
5.2	Clustering performance on the BBCSport, MSRCv1, Caltech101-7, Extended-Yale, MNIST and MNIST-1000 datasets.	79
5.3	Method comparison on COIL20 and Out-Scene datasets.	83
5.4	Time cost (s) of MVCSK, NESE and MKGNE algorithms respectively.	84
6.1	Clustering performance on the ORL, Outdoor-Scene and Coil20 datasets.	94
6.2	Clustering performance on the BBCSport, MSRCv1, Extended-Yale, MNIST and MNIST-1000 datasets.	95
6.3	Clustering performance on the three synthetic datasets.	96
6.4	Ablation study with different models.	96
7.1	Clustering performance on the COIL20, ORL, and Out-Scene datasets. The best performance for each indicator is in bold.	106
7.2	Clustering performance on the MSRCv1, BBCSport, Extended-Yale, and MNIST-10000 datasets.	107
7.3	Clustering performance on the Caltech101 dataset. The best performance for each indicator is bolded. The "-" symbol indicates that the ARI indicator was not provided by the corresponding published paper.	107
7.4	Ablation study with different conditions. The best performance for each indicator is in bold.	109
7.5	The time cost (seconds) of MVCSK, NESE, and CNESE.	112

8.1	Clustering performance on the COIL20, ORL, Out-Scene, and NUS datasets.	122
8.2	Clustering performance on the MNIST-10000 and the MNIST-25000 datasets.	123
8.3	Ablation study with different conditions.	123
9.1	Clustering performance on the COIL20, ORL, and Out-Scene datasets.	142
9.2	Clustering performance on the BBCSport, MSRCv1, Handwritten, Extended-Yale and MNIST-10000 datasets.	143
9.3	Ablation study with different models. The best performance for each indicator is in bold.	143

Notations

$\mathbf{X}^v = (\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v)$	Data matrix of the v -th view $\in \mathbb{R}^{d^v \times n}$
\mathbf{x}_i^v	The i -th column of \mathbf{X}^v
$\ \mathbf{X}\ _F$ or $\ \mathbf{X}\ _2$	Frobenius or ℓ_2 norm of the matrix \mathbf{X}
X_{ij}	Element of i -th row and j -th column of a matrix \mathbf{X}
\mathbf{K}^v	Kernel matrix of the v -th view $\in \mathbb{R}^{n \times n}$
\mathbf{S}^v	Similarity matrix (graph matrix) of the v -th view $\in \mathbb{R}^{n \times n}$
\mathbf{S}^*	Consensus similarity matrix (graph matrix) $\in \mathbb{R}^{n \times n}$
\mathbf{D}	Degree matrix (diagonal matrix)
\mathbf{D}^v	Degree matrix of the v -th view
\mathbf{L}^v	Laplacian matrix of the v -th view $\in \mathbb{R}^{n \times n}$
\mathbf{L}^*	Consensus Laplacian matrix $\in \mathbb{R}^{n \times n}$
\mathbf{P}^v	Spectral projection matrix of the v -th view $\in \mathbb{R}^{n \times C}$
\mathbf{P}^*	Consensus spectral projection matrix of the v -th view $\in \mathbb{R}^{n \times C}$
\mathbf{H}	Nonnegative embedding matrix (soft cluster assignments) $\in \mathbb{R}^{n \times C}$
\mathbf{I}	Identity matrix
$\mathbf{1}$	Column vector with all elements equal to one
$\text{Tr}(\cdot)$	Trace operator
n	Number of samples
C	Number of clusters
V	Number of views
d^v	Dimension of the feature vector of the v -th view
K	number of nearest neighbors to represent each data
$\beta, \mu, \gamma, \alpha, \lambda, \lambda_1, \lambda_2, \lambda_3$ and λ_4	Balance parameters

General Introduction to multi-view learning

1.1 General Introduction

In the past, machine learning faced many challenges for several reasons. First, the lack of training data was a problem for many researchers. In addition, there was the low computational power of computers, the weaknesses and limitations of existing learning algorithms, and the difficulties in processing and handling large data sets. Nowadays, thanks to technological progress, a variety of data of different sizes and types can be collected from different fields such as medicine, finance, banking, etc., so the field of machine learning is experiencing many advances and the machine can achieve high performance.

Therefore, the use of machine learning (ML) has become a mandatory task to process the data and understand the hidden patterns in the data, which makes the performance of the machine more efficient. Basically, three types of learning are used by these machines to process data. These types of learning are distinguished by the availability of data labels, which can be considered as the specific information of each data set that categorizes each group or cluster. The first type concerns supervised learning, where the labels of the data are known and used during the learning step. The second type is unsupervised learning, where the model works independently to discover and analyze the data (e.g., to find groups with unknown patterns in the data). During the learning process, there is no further information about the labels of the data. The third type of learning is semi-supervised learning, where only a small number of data instances are labeled.

When it comes to categorizing data, the goal of all these types of learning is to divide data into different groups so that data in the same group are similar to each other and data in different groups are

dissimilar.

Unsupervised learning is adopted in this thesis. Unsupervised learning approaches can be extremely useful for the field of Big Data analytics. These algorithms process data without knowing information about its category. A typical and widely used unsupervised learning task is clustering. Clustering divides data into multiple groups based on the similarity between data points. For example, in biology, clustering can be used to identify genes with similar functions. In addition, in information retrieval, clustering can facilitate and organize the result of a query. Thus, the result of a query for "movie" provides several categories divided into "drama", "love", "romance", "action", "comedy", etc. Fig. 1.1 shows an example of clustering.

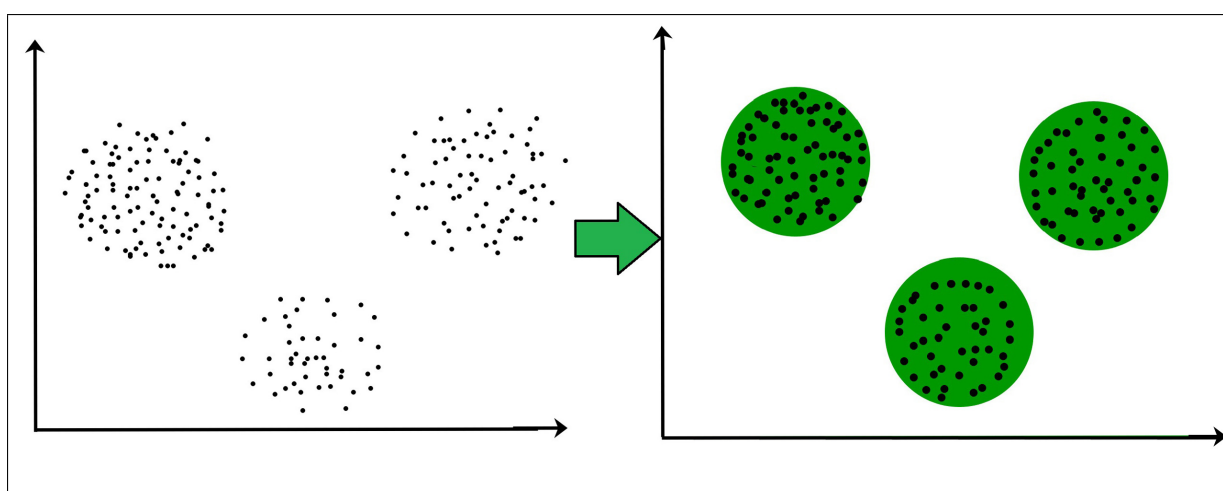


Figure 1.1: Disordered data before clustering (left) and ordered after clustering (right) [1].

Clustering can also be used in various fields:

- Marketing: Clustering is used in marketing to characterize and discover customer segments for marketing purposes.
- Biology: It can be used to identify different groups of plants.
- Libraries: It can be used to cluster books by their topics and content.
- Medicine: It can be used to identify different stages of diseases.
- Earthquake Studies: It can be used to divide areas into different zones.
- Also, clustering is used for insurance, urban planning, etc.

Nowadays, it is possible to have multiple representations or views for the same dataset. Therefore, clustering multiple views has attracted much attention in machine learning. For example, a web page

can be represented either by its content, by its title, or by its hyperlink. Another example is an article, which may be written in English, French, Spanish, etc. In addition, a person can be characterized by multiple images, each taken in a particular position. In other words, multi-view data is a group of features collected from different sources. They allow better use of the data and provide additional information about the data to improve clustering results. Although the use of multiple views can make the approaches more complicated due to the huge amount of additional information, in general, the use of these views can be considered as a good way to achieve better clustering results, as they provide additional information, especially when the information of a particular view is not sufficient to achieve the desired target clustering.

Since the principle of clustering is essentially based on the creation of a similarity matrix between the different data points, an algorithm capable of creating a precise and consistent similarity matrix must be developed.

Motivated by the great advances in the methods used for clustering, in this thesis we have proposed several algorithms capable of overcoming the challenges of the already published methods, in addition to their mathematical models. The desire to develop suitable and discriminative algorithms for multi-view clustering is an important task for researchers. A crucial step in multi-view clustering approaches is to learn appropriate data representations to extract meaningful information from the dataset. Although significant progress has been made in learning graph-based multi-view clustering, more research is needed. Indeed, most multi-view clustering approaches have a number of limitations in terms of the quality of the extracted information from the dataset. This thesis contributes to unsupervised learning by using different models capable of performing multi-view clustering. Different multi-view learning algorithms and their applicability to different types and sizes of datasets have been the focus of our research. Most of the experimental results obtained in testing the proposed methods have shown that they are superior compared to many powerful competing methods. Matrix factorization is used to create a soft clustering assignment matrix in most of our presented methods, which allows these methods to improve their performance.

1.2 Benefits of multi-view Learning

The three main advantages of multi-view learning are explained below.

1) Providing complementary information to generate a complete pattern of the data: Multi-view data contribute to the discovery of a complete "dataset". Single-view data often contain incomplete information about the dataset, while multi-view data usually contain sufficient detail. By gathering

complementary information from multi-view data, a full pattern of the dataset could be generated, especially when the deficiencies of one view are completed by the information of other views. In other words, the multi-view methods allow to take advantage of the additional information contained in the different views to obtain an accurate description of the dataset in order to obtain the best clustering results.

2) Noise Reduction: Learning from multiple views can reduce the noise in the obtained dataset, i.e., the multi-view methods can overcome the problem of noise which, once it affects the single view, it requires high cost to avoid poor clustering results.

Fig. 1.2 shows an example of how multi-view data can reduce the effects of noise. As shown in this figure, an image is examined using four different cameras. Due to the general limitations of imaging technology, all of these photos look very noisy, which could make further analysis difficult. By minimizing the overall noise for each individual view, this example proves how learning from multi-view data contributes to strong results. In general, the presence of noise in a single view makes it difficult to detect a particular pattern, resulting in unsatisfactory analysis of single view data. However, with multi-view learning, we can avoid the effect of noise in each individual view by highlighting the common pattern among the different views.

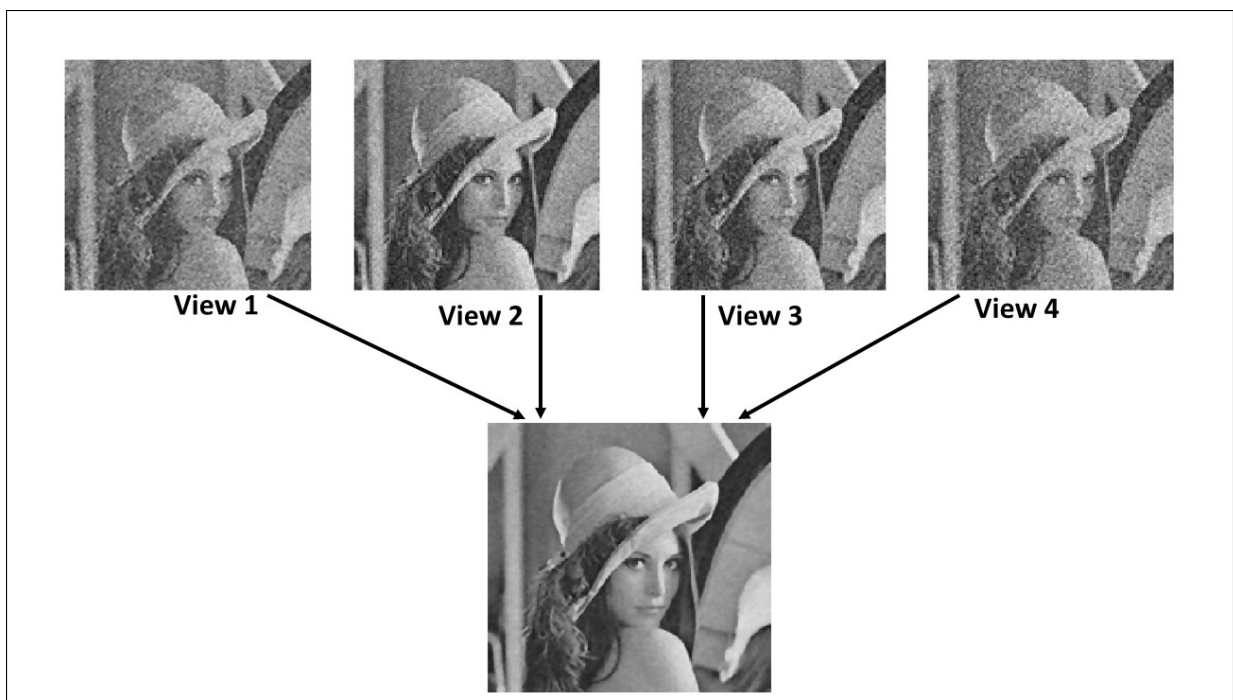


Figure 1.2: Reducing the effect of noise.

3) Applicable to a large number of applications: All multi-view clustering approaches may, without any doubt, be applied to single-view data. Unfortunately, single-view clustering methods are

not able to directly perform multi-view clustering applications due to their limitations. For instance, multi-view data are becoming more common. These forms of data are best suited for multi-view learning, while single-view learning approaches cannot adequately resolve them. Overall, the complementary nature of multi-view data can help reduce the shortcomings of single-view data and increase their applicability.

1.2.1 Practical problems of multi-view learning approaches

Although multi-view clustering methods have become increasingly important, these methods cannot fully solve some of the most important research problems in the field. Therefore, it is important to address the challenges of these approaches.

First, multi-view data come from different feature domains. To enable joint analysis, data from multiple views must be described in a unified way. A fundamental difficulty in learning from multiple views is how to properly handle multi-view data. Therefore, it is imperative to develop a method that can take advantage of multiple views by exploring the underlying relationships between them. Although different single-view data contribute to learning from multiple views in different ways, leveraging their combined effects to improve learning tasks remains a challenging issue. The third challenge is to highlight the importance and contribution of each view in clustering. Also, multiple views have the ability to rapidly increase the amount of data. Preprocessing multi-view data by dimensionality reduction or feature selection seems to be a necessary step for subsequent analysis. In this thesis, we attempt to develop efficient methods that can overcome these limitations. Our contributions are listed below.

1.3 Research outline and manuscript structure

In this thesis report, we have proposed several graph-based multi-view approaches that have produced remarkable results and outperform many existing methods. We have also provided a brief review of different types of machine learning. We also present numerous examples to illustrate these concepts with their advantages, disadvantages, and variants.

This thesis consists mainly of our own and original contributions. All chapters include our individual contributions to multi-view clustering analysis and algorithmic innovation. We developed and tested the proposed techniques on the experimental datasets, and analyzed the results. Moreover, our proposed methods were applied to different types and sizes of datasets, not only images, demonstrating the superiority of our methods.

The main contributions of our thesis are presented below.

- **Direct Multi-view Spectral Clustering with Consistent Kernelized Graph and Convolved Nonnegative Representation (Chapter 5):** In this chapter, a novel approach called Multi-view Clustering via Kernelized Graph and Nonnegative Embedding (MKGNE) is discussed. It can jointly provide (i) the unified similarity matrix across all views using the kernel tricks, (ii) the unified spectral projection of the data, (iii) the nonnegative cluster index matrix, and (iv) the weight of each view without additional parameters. Our proposed method takes the different kernel matrices corresponding to the different views as input. By creating a nonnegative embedding matrix, no post-processing such as k-means or spectral rotation is required. Several experiments conducted on real datasets demonstrate the effectiveness of the proposed method.
- **Consensus graph and spectral representation for one-step multi-view kernel based clustering (Chapter 6):** In this chapter, a new method called Multi-view Clustering via Consensus Graph Learning and Nonnegative Embedding (MVCGE) is presented. This method is similar to the MKGNE method presented in Chapter 5. Thus, the consensus affinity matrix (graph matrix), consensus representation, and cluster index matrix (nonnegative embedding) are learned simultaneously in a unified framework. Besides, an orthogonality constraint is imposed on the nonnegative embedding matrix used as the cluster indices matrix, which should also be as close as possible to the graph convolution of the consensus representation. Another difference between the MKGNE method and our new model is that our proposed approach integrates a smoothness constraint on the nonnegative embedding matrix \mathbf{H} (soft cluster assignments) over the graphs. Our method is tested on real and synthetic datasets. The experiments performed show that the proposed method performs well compared to many state-of-the-art approaches.
- **Multi-view Spectral Clustering via Constrained Nonnegative Embedding (Chapter 7):** In this chapter, a new multi-view spectral clustering via constrained nonnegative embedding (CNESE) is presented which can be considered as an improved version of the method "Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding" (NESE) [11]. Besides retaining the advantages of the NESE method, our proposed model integrates two types of constraints: (i) a consistent smoothness of the nonnegative embedding over all views, and (ii) an explicit orthogonality constraint over the columns of the nonnegative embedding matrix. Experimental results on several real datasets show the superiority of the proposed approach.
- **One-step Multi-view Spectral Clustering with Cluster Label Correlation Graph (Chapter 8):** In this chapter, a novel approach called Multi-view Spectral Clustering with a Self-taught

Robust Graph Learning (MCSRGL) is proposed. The main novelty of this method is the use of two types of graphs to perform multi-view clustering. In the first type, the multiple similarity graphs are constructed considering the similarity in the data space in each view. The second type of graph, called the cluster label correlation graph, represents the graph of the label space. Second, a smoothing constraint is used to constrain the cluster-label matrix and make it more consistent with the original data graphs as well as with the label graphs. Experimental results on several public datasets show the efficiency of the proposed approach.

- **A Unified Framework for Multi-view Clustering via Consensus Graph and Spectral Representation Learning (Chapter 9):** In this chapter, two novel approaches are proposed. The first approach is called One Step Multi-view Clustering via Consensus Graph Learning and Soft Clustering Assignments (OSMGSCA). This method involves learning the individual graphs as well as a consensus graph that are friendly to clustering, and thus can reduce the problem of merging different graphs that can contain noise. This method can simultaneously provide the consensus similarity matrix, the similarity matrix of each view, the corresponding spectral projection matrix of each view, the nonnegative cluster indicator matrix, and the weight of each view automatically. In addition, we present an improved version of our method that computes and relies on the consensus graph and consensus spectral projection matrix instead of relying on the individual matrices for each view. This method is called Unified Multi-view Clustering via Joint Graph Learning and Nonnegative Matrix Assignments (U-MCJGLNMA). Extensive experiments conducted on several real-world datasets of different types and sizes show the superiority of our two methods compared to other state-of-the-art methods.

More details of such contributions are included in their corresponding chapters.

The thesis is organized as follows. It is based on a general introduction, followed by our contributions. This chapter provides a general overview of the dissertation and discusses the research outline and the manuscript structure. The second chapter introduces and discusses the basic machine learning techniques. Related work is presented in Chapter 3. Chapter 4 describes the experimental setup of our methods. In the remaining chapters of the thesis, each contribution is described in detail in a separate chapter. Chapter 10 concludes the thesis. Finally, Chapter 11 presents the main papers and publications produced during the course of the thesis.

General Introduction to machine learning concepts

2.1 Different types of machine learning methods

Machine learning has faced many difficulties in the past due to a number of factors. These include the lack of training data, the low computational power of computers, the shortcomings and weaknesses of various learning algorithms, and the complexity of handling and processing large data sets. Thanks to the increasing computational power of computers and the availability of huge data sets that can be used for training, the field of machine learning has made many advances in recent years that enable the machine to achieve great performance. The more abundant the data, the better the machine can learn. To process data, these machines use a variety of learning techniques. There are three different types of learning. The first category is supervised learning, where the data labels are known. The second category is unsupervised learning, in which the model itself discovers and processes information (e.g., finding groups of unknown patterns in the data). Semi-supervised learning is the third method of learning, in which some data are labeled while others are not. A detailed description of these three types of learning methods can be found below.

Supervised Learning

Supervised learning, commonly referred to as supervised machine learning, is a subset of artificial intelligence and machine learning. In this type of learning, labeled data sets are used to train various models that efficiently classify data or predict outcomes. Once the data is fed into the model, the latter modifies its weights and will be properly fitted, which happens in the cross-validation phase. This type of learning can be used to solve a number of real-world problems. In supervised learning, a training data

set is used to train the classifiers to produce the specified output. The loss function is used to evaluate the computational accuracy, and it is adjusted until the error is satisfactorily reduced. Supervised learning is divided into two main categories: Classification and Regression.

- **Classification:** A classifier is used to classify test data and assign it to a particular group. It identifies certain patterns in the data set and makes accurate predictions about how these elements should be classified or labeled. Support Vector Machines (SVM), linear classifiers, k-nearest neighbor, decision tree and random forest are some of the most popular classifiers.
- **Regression:** To study the correlation between two variables, regression is often used. The best-known regression models include logistic regression, linear regression, and polynomial regression.

Some supervised learning methods:

In supervised classification, a variety of methods and processing approaches can be used. The following includes some of the most commonly used learning algorithms.

- **Naive Bayes:** The Naive Bayes classification technique is characterized by the application of the principle of Bayes theorem in terms of conditional independence of class. This means that the presence of one element does not affect the presence of another element in the probability of a particular event. Then, each prediction has an equivalent effect on the outcome. The three main types of Naive Bayes classifiers are Bernoulli Naive Bayes, Gaussian Naive Bayes, and Multinomial Naive Bayes. Text classification, spam detection, and recommendation systems use this approach.
- **Linear Regression:** Linear regression is a statistical technique for determining the relationship between a dependent variable and one or more independent variables and is often used to estimate future outcomes. Simple linear regression is used when there is only one independent variable and one dependent variable. Multiple linear regression is used when the number of independent variables is increased. It aims to create a boundary of maximum fit, which is determined by using the least square method. In a graph, this line is straight, which is different from other regression models.
- **Logistic regression:** While the linear regression method is used when the dependent variables are continuous, logistic regression is used when the dependent variables are categorical, such as "true" and "false" or "cat" and "dog". Although both regression methods aim to discover correlations between input data, logistic regression is more commonly used for binary classification tasks such as spam detection.

- **Support Vector Machine (SVM):** The famous Support Vector Machine (SVM) is a common supervised learning method developed by Vladimir Vapnik that can be used for both classification and regression tasks. However, it is most commonly used for classification tasks, where it creates a hyperplane where the separation between the two categories of data sets is greatest. The decision boundary is simply a hyperplane that separates the categories of data sets on each side of the plane.
- **K-nearest neighbor (K-nn):** The K-nearest neighbor (K-nn) is a nonparametric method that classifies data inputs according to their proximity and relationship to other data points. This technique implies that data points with comparable characteristics could be in close proximity to each other. Consequently, it attempts to determine the distance between different data points, usually using Euclidean distance, and then assigns a label based on the most frequent class or average. This method is characterized by its simplicity and fast computation time, but as the test data set becomes larger, the computation time increases, making it somewhat less suitable for classification problems.
- **Random Forest (RF):** The Random Forest (RF) is another famous supervised machine learning technique that can be used for regression and classification problems. The "forest" refers to a group of independent decision trees that are combined to reduce variance and achieve a better prediction.

Semi-supervised learning methods:

Semi-supervised learning is a type of machine learning that occurs when some of the observations in the training dataset are not yet labeled, i.e., in this type of learning, a classifier should learn and make predictions about new data by learning from a limited number of labeled instances and a large number of unlabeled instances. Semi-supervised learning is therefore a task that lies somewhere between supervised and unsupervised learning methods.

Unsupervised learning methods:

"Development of unsupervised pattern recognition methods is the most challenging and promising area of research today." Andrew Ng, NIPS Conference, Dec. 2016

Unsupervised learning is a third type of machine learning method in which the model does not require user supervision. However, it allows the model to work independently to identify previously undiscovered patterns and information. It deals primarily with unlabeled data. Unsupervised learning algorithms have great potential in the field of data analysis. These algorithms can be very useful because

they have no knowledge about the labels of the data. Clustering is a common task of unsupervised learning. This thesis is mainly concerned with unsupervised machine learning approaches.

Characteristics of our multi-view clustering algorithms

Concatenating multiple features of the same data into a single feature vector is a useful concept for combining data from multiple views. Since the pattern of each view is different, it is not a good idea to treat numerous views as a single view by simply combining all features into one long feature vector. When combining data from multiple views into a single feature vector, the different views are processed in the same way and their differences are ignored. Therefore, instead of treating the data from multiple views as one combined feature vector, we treat these views as a set of different similarity matrices or kernels. Multiple graphs can be used to represent data with multiple views. Typically, a similarity matrix is used to describe each graph. We can develop a similarity matrix for each view and derive the sum of these similarity matrices for clustering multiple views by considering each view as different from the other views. Although similarity aggregation seems to be a simple and efficient implementation that even leads to good clustering results, an automated process that assigns different weight values to each similarity or kernel matrix is still preferable. Such a weighting approach can contribute to additional benefits, such as removing or reducing the impact of noisy views. We perform multi-view clustering using different views of the same data. The goal is to properly combine the data from multiple views and use their intrinsic relationship to simplify the clustering task. The typical k-means clustering algorithm is explained below.

K-means clustering algorithm

One of the most basic and widely used unsupervised machine learning algorithms is k-means clustering. The goal of the k-means method is to partition a set of observations into a predetermined number of k clusters. Its main steps are summarized below.

First, a certain number of clusters k is specified, which corresponds to the number of centroids needed in the data set. The centroid is a point that represents the center of each defined cluster. Each cluster in k-Means is associated with a centroid. The goal of k-means clustering is to minimize the inter-cluster sum of squares, i.e., the sum of distances between different data points and their corresponding cluster centroid. Each data point is then assigned to one of the defined clusters.

Steps of the k-means clustering algorithm:

The k-means clustering technique starts by randomly selecting a certain number of centroids (μ) to serve as starting points for each cluster, and then performs iterative computations to improve the placement of the centroids. All observations \mathbf{x}_p are assigned to the nearest centroid $\mu_i^{(t)}$ at each iteration

t of the algorithm (see Eq. 2.1). If multiple centroids are at the same distance from a given data point, one centroid is randomly selected.

$$\mathbf{S}_i^{(t)} = \{\mathbf{x}_p : \|\mathbf{x}_p - \mu_i^{(t)}\|^2 \leq \|\mathbf{x}_p - \mu_j^{(t)}\|^2 \forall j, 1 \leq j \leq k\}, \quad (2.1)$$

where $|\mathbf{S}_i^t|$ is the total number of samples \mathbf{x}_j associated with the centroid $\mu_i^{(t)}$. After that, by computing the mean of the assigned samples to the corresponding centroids, the centroids are adjusted (see Eq. 2.2).

$$\mu_i^{(t+1)} = \frac{1}{|\mathbf{S}_i^{(t)}|} \sum_{\mathbf{x}_j \in \mathbf{S}_i^t} \mathbf{x}_j. \quad (2.2)$$

The algorithm stops creating and updating clusters when:

- The centroids of the newly created clusters remain the same.
- The points remain in the same cluster.
- The specified number of iterations has been completed.

The k-means clustering technique attempts to optimize the objective function described in Eq. (2.3).

The procedure usually ends at a local minimum, since there is a fixed number of iterations to reach the various assignments for given centroids. Moreover, each step must lead to a better solution.

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2 \quad (2.3)$$

$$\text{with } r_{nk} = \begin{cases} 1 & x_n \in \mathbf{S}_k \\ 0 & \text{otherwise} \end{cases}$$

Advantages and disadvantages of the k-means clustering algorithm:

Advantages of the k-means clustering algorithm:

- Its implementation is really simple.
- It can handle a large amount of data and respond quickly to big data.
- It responds quickly to new cases.
- It ensures convergence.

- Cluster adaptation to different sizes and shapes.

Disadvantages of the k-means clustering algorithm:

- It is highly sensitive to noise and outliers.
- It is highly dependent on initial values.
- Manual selection of the number of clusters is a difficult task.
- Its scalability decreases with increasing dimensionality.

The fundamental problem with k-means is the dependence of the obtained results on the initially chosen centroids. If one of the defined centroids is more attracted to noise or outliers, similar data points may be clustered into different groups, while other, more distant data points are clustered together. The simplest solution to this problem is to repeat the clustering procedure, using different starting positions of the centroids. Then, the clustering result that appears most frequently is considered appropriate. To overcome the limitations of the classical k-means algorithm, the popular Spectral Clustering (SC) algorithm can also be used. This algorithm projects the data set into a space where it is linearly separable. In the rest of this chapter, the famous SC algorithm is described in detail after explaining the concept of graph-based methods.

Graph-based methods

Recently, graph-based multi-view clustering methods have attracted much attention in the fields of pattern recognition and machine learning. Various graphs are used to represent data with multiple views. These approaches work with a graph in which a node represents a data point and a weighted edge connects two nodes. Graph-based methods are currently widely used in a number of disciplines, such as graph-based embedding, semi-supervised learning, feature selection, regression, and clustering, where the graph represents the pairwise similarities between data points.

Given a collection of n data samples, represented by $\mathbf{X}^v = (\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v) \in \mathbb{R}^{n \times d^v}$, where d^v is the dimension of the feature vector in the v -th view of the data. We define the graph $G = (V; E; \mathbf{W})$, where V is the total number of nodes or vertices ($|V| = n$), E is the number of edges, and \mathbf{W} is the affinity matrix, also called the edge weight matrix. The edge weight between \mathbf{x}_i and \mathbf{x}_j , is determined by w_{ij} , which represents the similarity or distance between data points \mathbf{x}_i and \mathbf{x}_j . The affinity matrix \mathbf{W} is generally bound by the following constraints:

- $w_{ij} = 0$ means that there is no edge between data points \mathbf{x}_i and \mathbf{x}_j .

- $w_{ii} = 0, i = 1, \dots, n$ where n denotes the total number of data points or nodes.
- \mathbf{W} is a symmetric matrix: $w_{ij} = w_{ji}$.
- All weight edges are nonnegative $w_{ij} \geq 0$.

Currently, the most common method for creating an affinity matrix involves two steps: The first step is to create the adjacency matrix and the second step is to assign the edge weight matrix. A given set of data points can be transformed into a graph using one of the numerous widely used techniques. The purpose of creating similarity graphs is to describe the neighborhood relationships between different data points.

Some of the best traditional graph construction methods are the ϵ -neighborhoods graph, the K -nearest neighbors graph (K -nn), and the fully connected graph. The description of these methods can be found below.

- **ϵ -neighborhood graph:** The connection between two data points \mathbf{x}_i and \mathbf{x}_j is given if the pairwise distance between them is less than ϵ . A typical problem that may occur when using this technique is the probability of obtaining some disconnected nodes, which may occur when the value of ϵ is not properly fixed. Therefore, K -nearest neighbor (K -nn) graphs have been used to circumvent the above limitation.
- **K -nearest neighbors graph:** Using this graph, one can connect node \mathbf{x}_i to node \mathbf{x}_j if \mathbf{x}_i is one of the k -nearest neighbors of \mathbf{x}_j . However, since the neighborhood connection is asymmetric, this approach leads to a directed graph. This graph can be made undirected in two ways. First, we can simply ignore the edge direction and connect \mathbf{x}_i and \mathbf{x}_j with an undirected edge if \mathbf{x}_i is one of the k -nearest neighbors of \mathbf{x}_j or \mathbf{x}_j is one of the k -nearest neighbors of \mathbf{x}_i . The resulting graph is called the k -nearest neighbor graph. Second, the graph can be made undirected in another way by connecting \mathbf{x}_i and \mathbf{x}_j if both \mathbf{x}_i and \mathbf{x}_j were one of the k -nearest neighbors of \mathbf{x}_j and \mathbf{x}_i , respectively. The resulting graph is called a mutual k -nearest neighbor graph. However, when working with large data sets and thus a large number of neighbors are available to build the graph, the drawbacks of these graphs become apparent. The graph in this situation requires a significant amount of computing power. Figure 2.1 depicts four independent nodes connected to their nearest neighbors via the K -nn graph and the ϵ -graph. This figure shows some disconnected components obtained by using the ϵ -graph and how the problem was solved using the K -nn method.

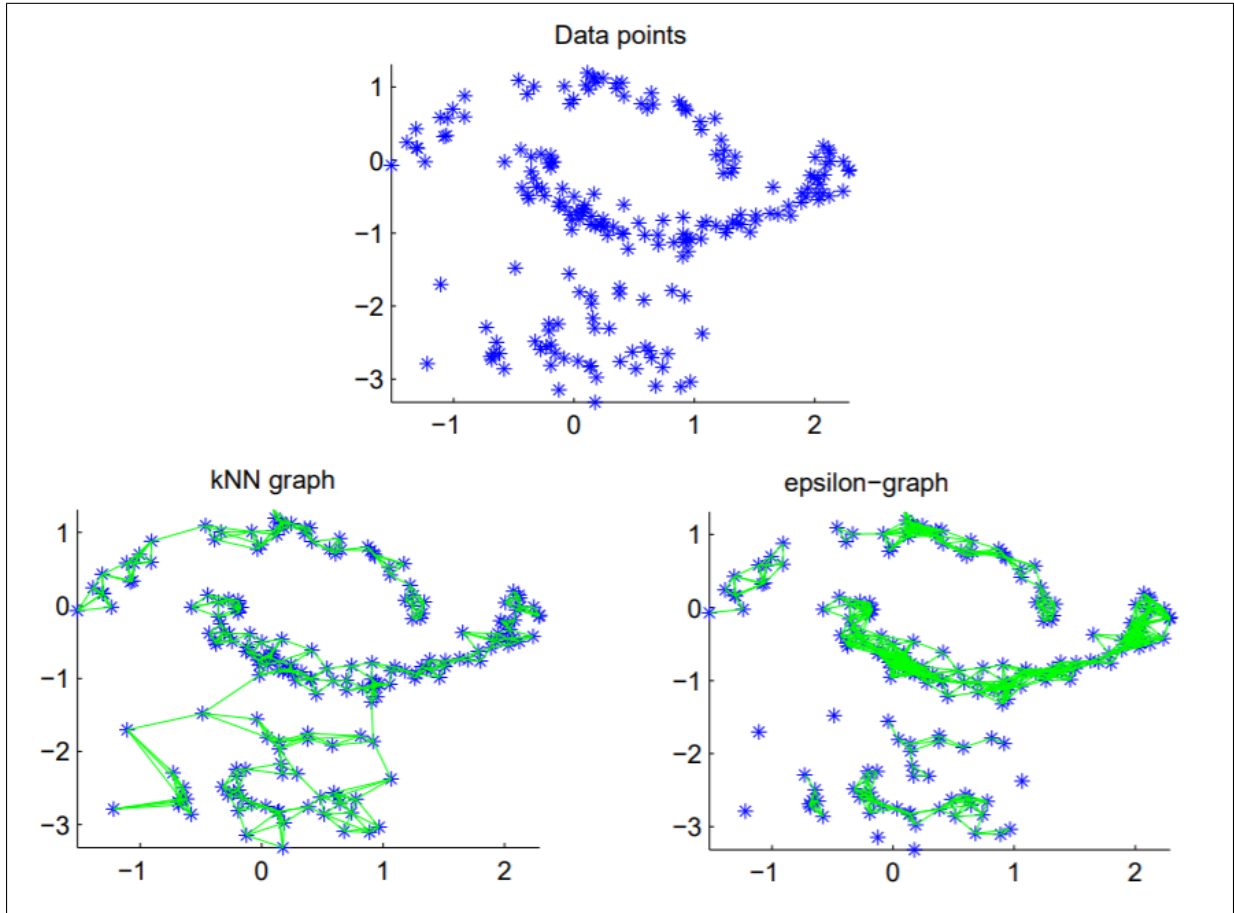


Figure 2.1: Different similarity graphs [2].

- **Fully connected graph:** By using this method, the graph is completely connected. Each node is connected to all other nodes in this graph. After computing the graph, the weight matrix is calculated to assign weights to the edges. Several methods can be used to construct the similarity matrix that represents this graph. A simple way to construct this matrix is to consider a binary weighting scheme.

$$w_{ij} = \begin{cases} 1, & \text{if there is an edge between the nodes } \mathbf{x}_i \text{ and } \mathbf{x}_j \\ 0, & \text{otherwise.} \end{cases} \quad (2.4)$$

However, this method treats all neighbors in the same way, regardless of their closeness or similarity. As a result, two nodes, no matter how close or far apart, receive the same value of weight.

Another similarity function can be used to overcome this limitation. As can be seen in Eq. (2.5), when a similarity function is used, each edge is assigned a weight based on the value of the similarity between the two nodes. Various functions can be used that serve as similarity measures. When two nodes are most similar, their edges receive a greater weight than dissimilar nodes.

Therefore, the influence of two nodes that are not close and are detected as neighbors by applying the K-nn algorithm can be suppressed using this strategy.

$$w_{ij} = \begin{cases} sim(\mathbf{x}_i, \mathbf{x}_j), & \text{if there is an edge between the nodes } \mathbf{x}_i \text{ and } \mathbf{x}_j \\ 0, & \text{otherwise.} \end{cases} \quad (2.5)$$

The Gaussian kernel function, which has the formula in Eq. (2.6) for weight computation, is now the most commonly used function.

$$sim(\mathbf{x}_i, \mathbf{x}_j) = -exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_j)}{T_0}\right), \quad (2.6)$$

where T_0 is the Gaussian kernel width and d is any distance measure, such as the Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|^2$. Besides, the cosine function can be used to compute the distance between \mathbf{x}_i and \mathbf{x}_j . A labeled, undirected graph with 6 nodes and its Laplacian matrix are shown in Figure 2.2. In this example, the similarity values are set to binary weights.

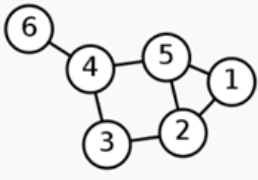
Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Figure 2.2: Undirected graph and its Laplacian matrix.

All of the previously described graphs are commonly used for spectral clustering. Now, we are going to explain the idea of the Laplacian of the graph, before moving on to explain the spectral clustering algorithm.

Graph Laplacian

Graph Laplacian matrices are the most commonly used features for spectral clustering today. The analysis of such matrices is the subject of an entire scientific discipline, spectral graph theory. We always consider that a graph G is usually assumed to be an undirected and weighted graph. Its weight matrix is represented by \mathbf{W} , where $w_{ij} = w_{ji} \geq 0$. Moreover, the spectral projection matrix \mathbf{P} is also computed. This matrix contains the eigenvectors of the Laplacian matrix \mathbf{L} . We should not always assume that the eigenvectors of the matrix are normalized. The eigenvalues must be sorted in ascending order. The eigenvectors related to the k smallest eigenvalues are called "the first k eigenvectors".

The definition of the unnormalized Laplacian matrix is given in equation (2.7).

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \quad (2.7)$$

In the following, we describe the main properties of the Laplacian matrix needed for spectral clustering.

- For each vector $\mathbf{f} \in \mathbb{R}^n$, we have: $\mathbf{f}' \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (\mathbf{f}_i - \mathbf{f}_j)^2$.
- \mathbf{L} is a symmetric matrix and semi definite positive.
- The constant vector $\mathbf{1}$ corresponds to the smallest eigenvalue of the matrix \mathbf{L} , which is 0, and the multiplicity k of the eigenvalue 0 is equal to the number of connected components in the graph.
- \mathbf{L} has exactly n positive eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

The Laplacian matrix can be normalized by the following equation.

$$\mathbf{L}_{normalized} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}. \quad (2.8)$$

Spectral Clustering

In the following, the concept of popular spectral clustering algorithms is explained in detail. Spectral clustering (SC) [12], is a famous clustering approach based on spectral graph theory. One of its special aspects is that the data set is grouped based on connectivity rather than distance, which allows concave clustering of samples. When processing and grouping this type of data, spectral clustering is among the most commonly used clustering approaches. Spectral clustering is indeed a matrix trace optimization problem. In this section, we show how well the spectral clustering algorithm can be used to solve problems with multiple views. The criteria by which a graph can be partitioned into non-connected subgraphs are defined by spectral graph theory. The spectral clustering algorithm is an example of a graph clustering algorithm, which is applicable to any kind of data described as a matrix representing the similarity between all data points.

Single-view spectral clustering algorithm:

We begin by considering the spectral clustering algorithm in a single-view case. Suppose that the spectral representation matrix is $\mathbf{P} \in \mathbb{R}^{n \times k}$, where n is the total number of observations and k is the total number of clusters. The concept of spectral clustering can be described as follows:

$$\min_{\mathbf{P}} Tr(\mathbf{P}^T \mathbf{L} \mathbf{P}) \quad s.t. \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (2.9)$$

The dimension is clearly reduced from $(n \times n)$ to $(n \times k)$ after obtaining the spectral representation matrix \mathbf{P} . Besides, projecting the raw data into another space allows us to cluster non-convex regions.

We suppose that our data consists of n data points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$, which can be arbitrary elements. The pairwise similarity between data points is calculated using a symmetric and nonnegative similarity algorithm, $s_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$, and the resulting similarity matrix is denoted by \mathbf{S} . The Spectral Clustering algorithm is summarized in **Algorithm 1**.

Algorithm 1 Spectral Clustering

Input: Data matrices $\mathbf{X} \in \mathbb{R}^{n \times d}$.

The similarity matrix \mathbf{S} .

The total number of clusters k .

Output: The clustering allocation for n data points in k different clusters.

Construct the weighted similarity matrix \mathbf{W} using one of the methods mentioned above.

Compute the corresponding Laplacian matrix \mathbf{L} .

Compute the smallest k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of \mathbf{L} .

Create the matrix $\mathbf{P} \in \mathbb{R}^{n \times k}$ containing the vectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.

Cluster the rows of the matrix \mathbf{P} into k different clusters using any post processing step such as k-means clustering.

Multi-view spectral clustering algorithm:

Clustering performance on inputs with multiple views could also be improved if the different views are properly integrated. In the context of spectral clustering analysis, there are several ways to integrate data from multiple views. Multi-view spectral clustering algorithms should be able to find the complementary information contained in the different views and thus produce better or more consistent clustering results. One possible solution is to create a unified similarity matrix extracted from the different views, and then apply traditional single-view spectral clustering to this unified similarity matrix.

2.2 Overview of Kernel methods

Kernels or kernel techniques (also known as kernel functions) are groups of different types of pattern analysis techniques. They have been used to solve non-linear problems.

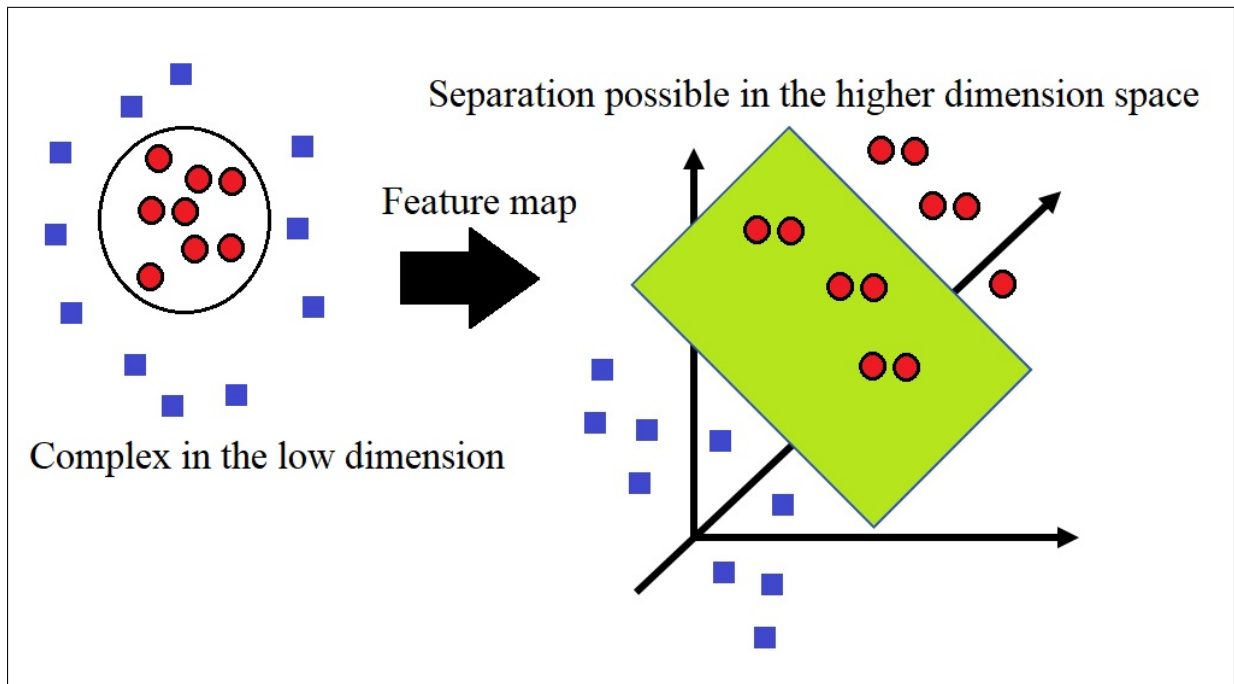


Figure 2.3: Feature map of the data set to a higher space.

From Figure 2.3, it is clear that using a linear classifier to accomplish this classification task is extremely difficult because there is no strong linear line that can categorize the red and blue points since the data points are randomly distributed. The kernel function is then used to transfer the data points into a higher dimensional space and generate the classification result. In other words: In the figure 2.3, the red points are inside a circular region and the blue data points are outside this region. So we transform a two-dimensional plane, previously categorized by a single line, into a three-dimensional space where our classification algorithm will be the hyperplane, a two-dimensional plane that divides the data set. The kernel trick, a way of using a linear classifier to solve a non-linear problem, is commonly referred to as a "kernel" in machine learning. It involves converting linearly inseparable data points into linearly separable data points (as shown in Figure 2.3). The special feature of the kernel trick is that the new space does not have to be explicitly declared. The data points are implicitly explored in another space by computing their dot products in this new space, as shown in equation (2.10).

$$K(\mathbf{x}, \mathbf{y}) = \langle f(\mathbf{x}), f(\mathbf{y}) \rangle, \quad (2.10)$$

where K is the kernel function used to map the data inputs \mathbf{x} and \mathbf{y} from the lower dimensional space n to the higher dimensional space m . The dot product between \mathbf{x} and \mathbf{y} is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. To get the value of the dot product $\langle f(\mathbf{x}), f(\mathbf{y}) \rangle$ we usually need to calculate $f(\mathbf{x})$, $f(\mathbf{y})$, and then perform the dot product. Since these two steps require calculations in m -dimensional space, where m can be a large number, they can be very expensive. Despite all our efforts to reach the high-dimensional space m , the

obtained dot product is a scalar. Therefore, we return to the one-dimensional space. Now the question arises, do we really need to do all this work just to get the value of this "scalar"? Is it really necessary to do these manipulations in m -dimensional space? The answer to this question is no, using the kernel trick. To illustrate this idea, an example is provided below.

Suppose we have two vectors, $\mathbf{x} = (1, 2, 2)$ and $\mathbf{y} = (1, 2, 3)$.

As mentioned earlier, to calculate the dot product between $f(\mathbf{x})$ and $f(\mathbf{y})$, we must first calculate $f(\mathbf{x})$ and $f(\mathbf{y})$.

Then: $f(\mathbf{x}) = (\mathbf{x}_1 * \mathbf{x}_1, \mathbf{x}_1 * \mathbf{x}_2, \mathbf{x}_1 * \mathbf{x}_3, \mathbf{x}_2 * \mathbf{x}_1, \mathbf{x}_2 * \mathbf{x}_2, \mathbf{x}_2 * \mathbf{x}_3, \mathbf{x}_3 * \mathbf{x}_1, \mathbf{x}_3 * \mathbf{x}_2, \mathbf{x}_3 * \mathbf{x}_3) =$

$(1, 2, 2, 2, 4, 4, 2, 4, 4)$.

$f(\mathbf{y}) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$.

$\langle f(\mathbf{x}), f(\mathbf{y}) \rangle = 1 + 4 + 6 + 4 + 16 + 24 + 6 + 24 + 36 = 121$

There are numerous calculations to be done because f is a transformation from a three-dimensional to a nine-dimensional space.

To solve this problem, the kernel trick can be applied as follows:

$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^2 = (1 + 4 + 6)^2 = 121$.

From the above example, it is clear that the result is the same, but the computation time is much easier by using the kernel trick.

In most of our methods, we use the kernel trick to represent the similarity between the different data points in a more appropriate space and to explore the nonlinear relationships between the different data points. Consider Φ as the mapping function from the original data space \mathbb{R}^D to the reproducing Hilbert space \mathcal{H} . Given the input data matrix for different views v : $\mathbf{X}^{(v)} = [\mathbf{x}_1^{(v)}, \mathbf{x}_2^{(v)}, \dots, \mathbf{x}_n^{(v)}]$, the Hilbert space transformation will be: $\Phi(\mathbf{X}^{(v)}) = [\Phi(\mathbf{x}_1^{(v)}), \Phi(\mathbf{x}_2^{(v)}), \dots, \Phi(\mathbf{x}_n^{(v)})]$.

A predefined kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ is used to represent the similarity between the two data points \mathbf{x}_i and \mathbf{x}_j . The kernel trick eliminates the need to know the exact transformation. As a result, calculations are greatly simplified. The most commonly used kernel methods are listed below.

- Linear Kernel: Given two input vectors \mathbf{x}_1 and \mathbf{x}_2 , the linear kernel is described as the dot product of these two vectors:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i * \mathbf{x}_j. \quad (2.11)$$

- Gaussian kernel: This non-linear kernel is commonly used in machine learning and has proven

to be a good choice for non-linear data. For a pair of data samples in a given view, the entry $K(\mathbf{x}_i, \mathbf{x}_j)$ is given by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2T_0 \sigma_0^2}\right), \quad (2.12)$$

where σ_0^2 is set to the average distance between pairs of samples in the considered view.

- Polynomial kernel: The polynomial kernel enables us to compute the high-dimensional relationship between the two observations. This type of kernel method is defined by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d, \quad (2.13)$$

where d and c are the degree and the coefficient of the polynomial respectively.

- Cosine kernel: In this type of kernel method, the normalized dot product between the two data points is calculated. If \mathbf{x}_i and \mathbf{x}_j are the two input vectors, the cosine similarity between them is calculated as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \mathbf{x}_j^T}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}. \quad (2.14)$$

- Sigmoid kernel: The sigmoid kernel between two data points, also known as hyperbolic tangent or multilayer perceptron, is computed using the sigmoid kernel function. This function can be used as a neuron activation function in the neural network model. It is specified as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + c_0), \quad (2.15)$$

where γ is known as the slope and c_0 is defined as the intercept.

- RBF kernel: The Radial Base Function (RBF) kernel between two data points is calculated with the RBF kernel function.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2). \quad (2.16)$$

If $\gamma = \frac{1}{\sigma^2}$, the RBF kernel is equivalent to the Gaussian kernel with variance σ^2 .

- Laplacian kernel: The Laplacian kernel is a type of the RBF kernel given by the following equation:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_1), \quad (2.17)$$

where $\|\mathbf{x}_i - \mathbf{x}_j\|_1$ is the Manhattan distance between the two input vectors. This type of kernel function is useful when the data contains noise.

Many other types of kernel functions can be used to calculate the similarity between data points. In our thesis, the Gaussian kernel is adopted.

2.3 Overview of Matrix Factorization

Matrix Factorization techniques are often used to reduce dimensionality. Clustering is performed on the generated coefficient matrix. These methods are characterized by their minimal computational requirements compared to other methods. Matrix factorization can be used to derive latent features from a data set when two different types of entities (i.e., two different matrices) are multiplied together. In other words, the given matrix is decomposed into the product of two smaller rectangular matrices using matrix factorization methods. Nonnegative matrix factorization (NMF or NNMF), also known as nonnegative matrix approximation, is a set of linear algebra techniques and a special type of matrix factorization in which a matrix \mathbf{X} is factorized into two distinct matrices \mathbf{Y} and \mathbf{Z}^T , all of which have no negative values.

2.3.1 Background

Given the matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, \mathbf{X} can be written as the product of two matrices: $\mathbf{Y} \in \mathbb{R}^{n \times k}$ and $\mathbf{Z}^T \in \mathbb{R}^{k \times m}$ as shown in Eq. (2.18).

$$\mathbf{X} = \mathbf{Y} * \mathbf{Z}. \quad (2.18)$$

The column vectors of \mathbf{X} can be computed as linear combinations or multiplications of the column vectors of the matrix \mathbf{Y} using coefficients provided by the columns of the matrix \mathbf{Z} . In other words, each column of the matrix \mathbf{X} can be calculated as follows:

$$\mathbf{x}_{i,:} = \mathbf{Y} * \mathbf{z}_{i,:}, \quad (2.19)$$

where $\mathbf{x}_{i,:}$ is the i -th column vector of the obtained matrix \mathbf{X} and $\mathbf{z}_{i,:}$ is the i -th column vector of the matrix \mathbf{Z} . The dimensions of each matrix used to form the matrix \mathbf{X} can be much smaller than those of the resulting matrix \mathbf{X} . This is the main feature of the NMF method. The NMF produces matrices with much fewer dimensions than the original matrix \mathbf{X} . For example, if \mathbf{X} is an $n \times m$ matrix, \mathbf{Y} is an $n \times k$ matrix, and \mathbf{Z}^T is a $k \times m$, k is smaller than n and m .

The following is an example that illustrates the concept of NMF.

- Consider that \mathbf{X} is an input matrix formed by 1000 rows and 50 columns of sentences and papers. That is, we have 50 papers with 1000 sentences classified. As a result, a paper is represented as a column vector \mathbf{x} in \mathbf{X} .
- Suppose we want the method to discover ten distinct features to generate the feature matrix \mathbf{Y} with 1000 rows and 10 columns and the coefficient matrix \mathbf{Z} with 10 rows and 50 columns.

- So, if we multiply the two matrices \mathbf{Y} and \mathbf{Z} , we get a matrix with 1000 rows and 50 columns, which has the same dimension as the matrix \mathbf{X} . Moreover, \mathbf{Y} and \mathbf{Z} are the nonnegative factorization of the matrix \mathbf{X} into two different matrices.
- As shown in the previous discussion of matrix multiplication, each column in the computed product matrix ($\mathbf{Y} * \mathbf{Z}$) is a linear combination of the 10 column vectors in the feature matrix \mathbf{Y} with coefficients supplied by the coefficient matrix \mathbf{Z} .

Since in our case we can assume that each paper is formed from a finite number of hidden features, this last assumption is the basis of NMF. Each component (represented by a column vector) in the feature matrix \mathbf{Y} can be thought of as a paper containing a group of sentences, where each cell value of the sentence defines its rank for a particular feature: The larger the cell value of a sentence, the higher its rank for that feature. A paper is represented by a column in the coefficient matrix \mathbf{Z} , where a cell value defines the rank of the paper for a particular feature.

As a result, a paper (column vector) can be created from our input matrix by using a linear combination of the column vectors in the matrix \mathbf{Y} with the cell value of the feature from the column of the matrix \mathbf{Z} .

2.3.2 Clustering property of the NMF method

The NMF has an inherent clustering capability, i.e., it automatically groups the columns of input data $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_2)$. More precisely, we look for \mathbf{Y} and \mathbf{Z} that optimize the objective function:

$$\|\mathbf{X} - \mathbf{YZ}\|_2, \text{ s.t. } \mathbf{Y} \geq \mathbf{0}, \mathbf{Z} \geq \mathbf{0}. \quad (2.20)$$

Furthermore, when an orthogonality condition is added to the objective function in Eq. (2.20) (i.e., $\mathbf{ZZ}^T = \mathbf{I}$), the minimization of the above optimization problem is equivalent to the optimization of the well-known k -means clustering algorithm.

The generated \mathbf{Z} , on the other hand, identifies the members of the cluster; for example, if $\mathbf{z}_{kj} \geq \mathbf{z}_{ij}$ for all $i \neq k$, the data point \mathbf{z}_j belongs to the k -th cluster. The matrix of cluster centroids is also computed and called \mathbf{Y} . In this matrix, the k -th column indicates the cluster centroid of the k -th cluster.

2.4 Overview of the Singular Value Decomposition algorithm

In linear algebra, the Singular Value Decomposition (SVD) of a matrix is a type of matrix factorization method that decomposes a given matrix into three lower dimensional matrices. The singular value

decomposition of a matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$, denoted by $SVD(\mathbf{X})$, is given by:

$$\mathbf{U}\Sigma\mathbf{V}^T = SVD(\mathbf{X}), \quad (2.21)$$

where $\mathbf{U}\mathbf{U}^T = \mathbf{V}\mathbf{V}^T = \mathbf{I}$. The k columns of the orthonormal matrices \mathbf{U} and \mathbf{V} represent the orthonormal eigenvector of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$, respectively. These columns are also called the left and right singular vectors of \mathbf{X} . The diagonal entries of the matrix Σ , which are the nonnegative square roots of the k eigenvalues of $\mathbf{X}\mathbf{X}^T$, are the singular values of the matrix \mathbf{X} . The singular values can be interpreted as the significance values of the various features in the matrix. Figure 2.4 illustrates the concept of the SVD method. As can be seen in Figure 2.4, thanks to the SVD decomposition, our original matrix can now be expressed as a linear combination of low-rank matrices, which is helpful in large computations.

2.5 Overview of the t-SNE method

The unsupervised nonlinear method called t-Distributed Stochastic Neighbor Embedding (t-SNE) is mainly used for data discovery and visualization of high-dimensional data sets. In other words, t-SNE provides an idea of how the data is distributed in a high-dimensional space. This method was developed by Laurens van der Maatens and Geoffrey Hinton in 2008. It converts multidimensional data into two or more dimensional spaces that can be easily processed and analyzed by humans.

The t-SNE method estimates the similarity distance between pairs of samples in high and low dimensional spaces. Then, using a cost function, it tries to maximize these two similarity measures.

2.6 Overview of the Gradient Descent method

Gradient descent is currently the most widely used optimization approach in machine learning. It can be used with any machine learning or deep learning algorithm. It is very easy to learn and implement. Anyone working with machine learning needs to be familiar with gradient descent concepts. Gradient descent is an optimization technique used in training or optimizing a particular algorithm or model in machine learning. This optimization method is based on a convex optimization problem where parameters are iteratively changed to minimize a given cost function to its local minima. First, the initial parameter values of the model are defined and then the algorithm uses mathematics to iteratively change these values to minimize the specific function. Moreover, a gradient is a measure of how the result of a function changes when the inputs are changed slightly. It also characterizes its slope. The larger the slope and the faster a model can be trained, the larger the gradient will be. However, if the

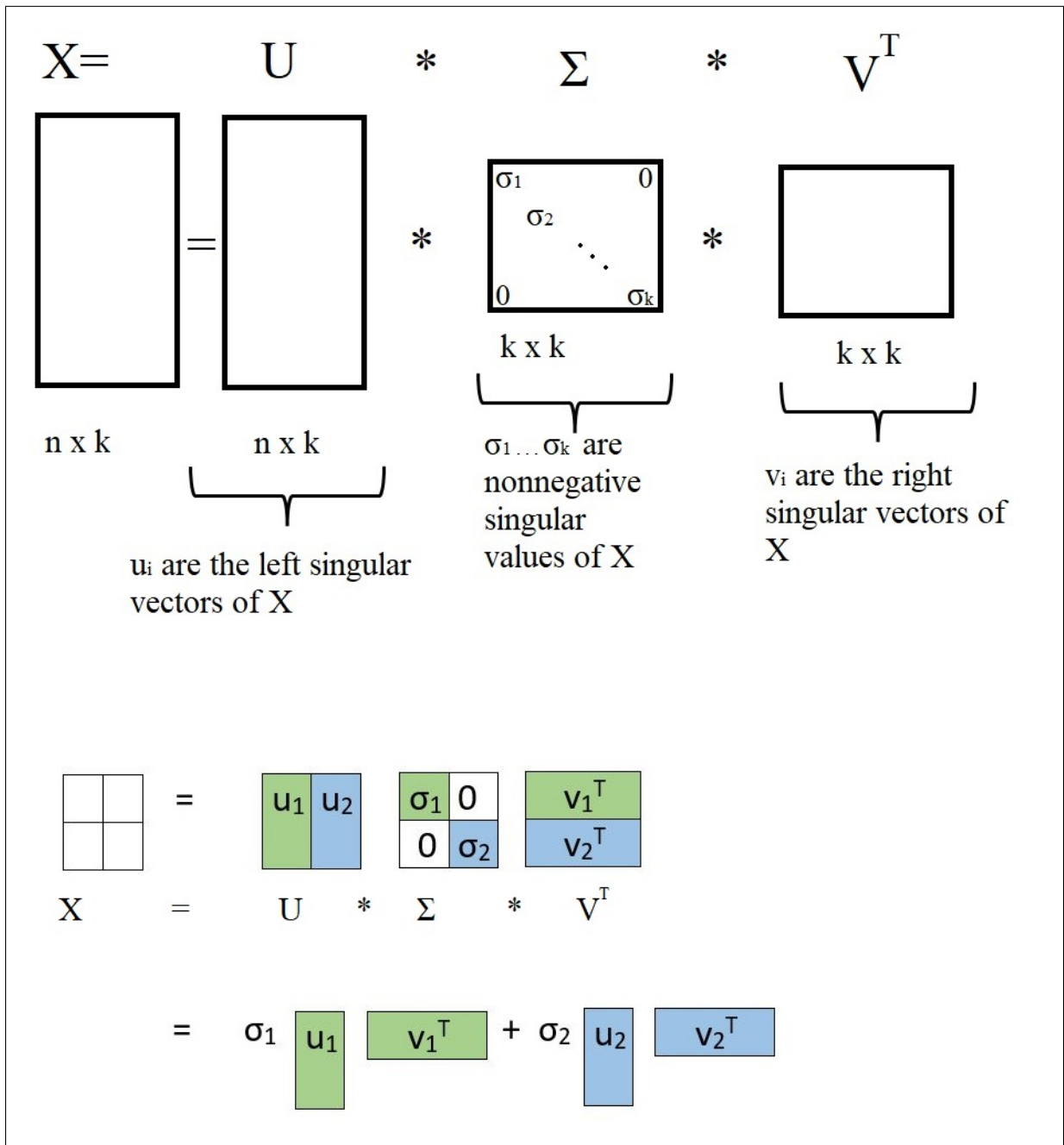


Figure 2.4: Illustration of the concept of the SVD method.

slope is zero, the model stops learning. In mathematics, a gradient can also be thought of as a partial derivative with respect to its inputs. Imagine a person trying to reach the top of a mountain with as few steps as possible. He could start climbing the mountain by taking very large steps on the shortest path, which he can do since he is not near the top of the mountain. However, as he approaches the summit, his steps become smaller over time to prevent him from exceeding the summit. The gradient could be used to properly characterize this behavior.

Consider Figure 2.6 as the horizontal plane of our mountain, with the blue arrows representing

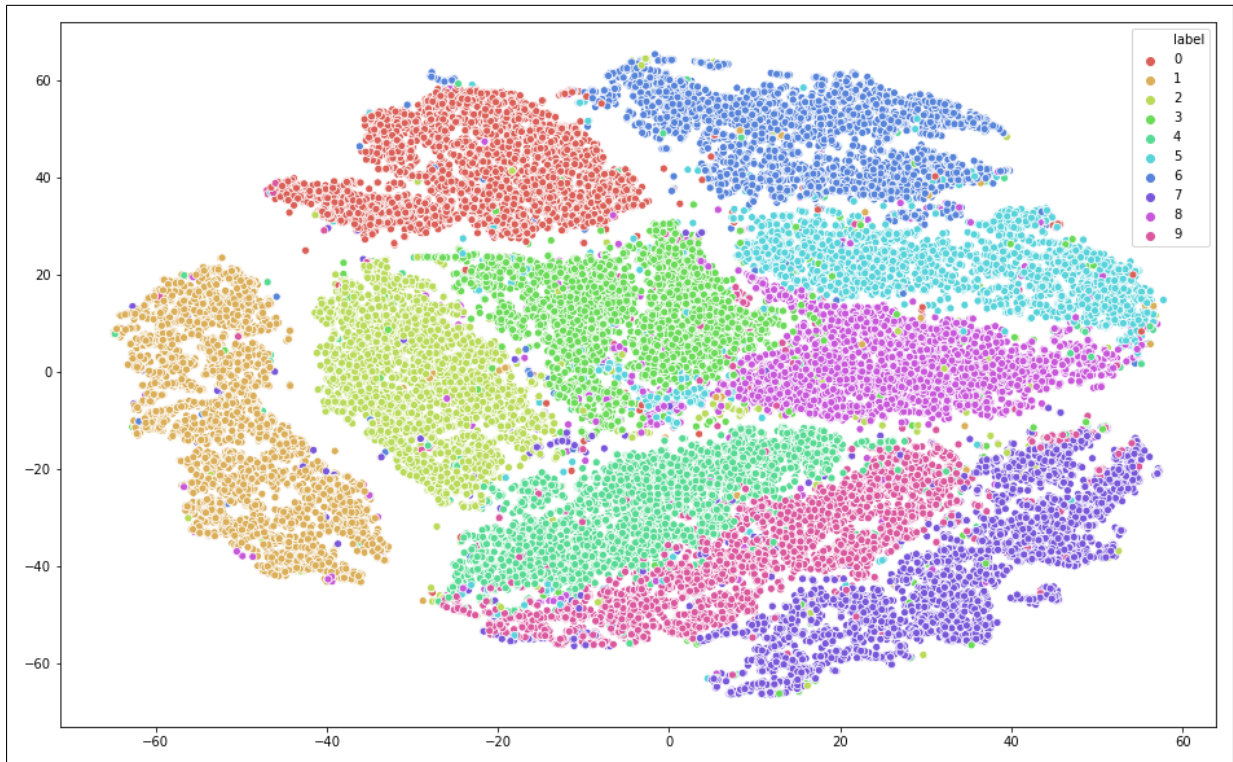


Figure 2.5: Illustration of the MNIST data set after applying t-SNE.

the steps of the person. The gradient in this case can be considered as a vector representing both the direction and the length of the best step the person could take.

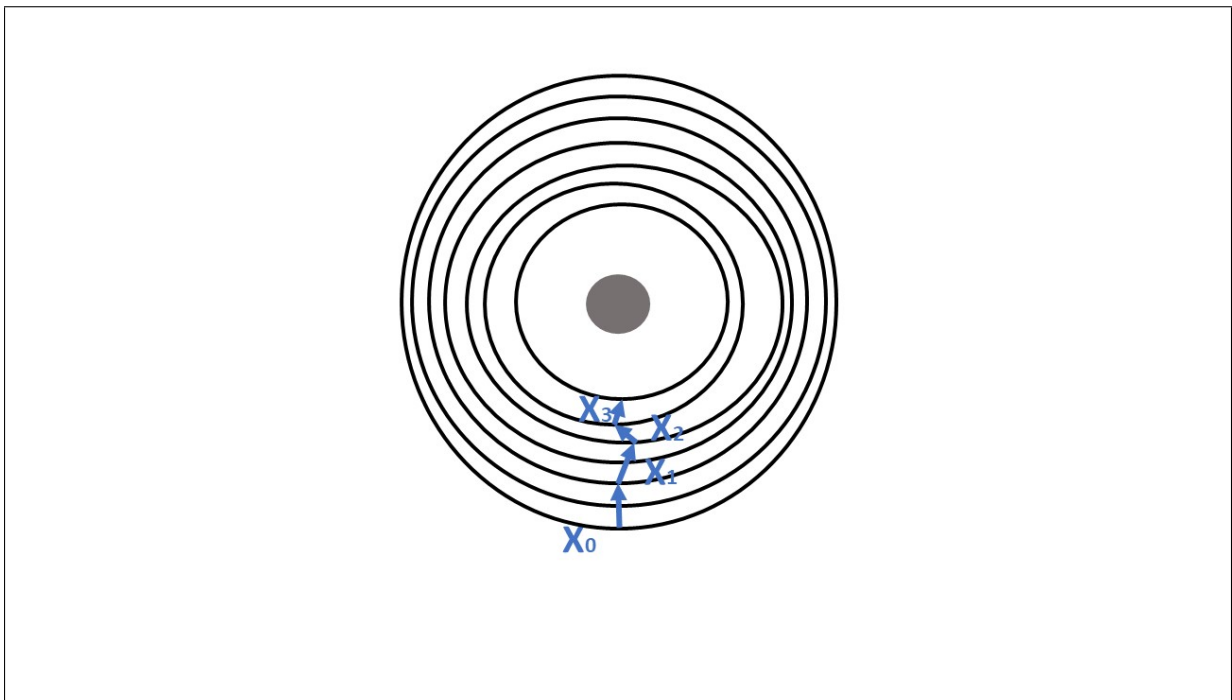


Figure 2.6: Illustration of the steps taken to reach the mountain from top to bottom.

It is worth noting that the gradient from \mathbf{X}_0 to \mathbf{X}_1 is much longer than that from \mathbf{X}_2 to \mathbf{X}_3 . This is

because the slope of the hill, which determines the length of the vector, is smaller. Therefore, a smaller gradient means a smaller slope and fewer steps for the climber.

2.6.1 How Gradient Descent works

Think of gradient descent as going down to the bottom of a river rather than ascending up a mountain. Since gradient descent is a minimization method that optimizes a particular function, this example is more logical.

Gradient descent is described by the Eq. (2.22).

$$b = a - \gamma(\Delta f(a)), \quad (2.22)$$

where b indicates the next position or location of the person, and a represents its current position. The negative sign represents the minimization process of the gradient descent. The parameter γ is a weight parameter, and the gradient term $(\Delta f(a))$ simply represents the steepest direction of descent.

This algorithm effectively informs us of the next position we should take, which is the direction of steepest descent. To illustrate this, an example of an optimization problem in machine learning is explained below.

We consider a supervised learning problem and need to train our optimization scheme (i.e., algorithm) using gradient descent to optimize the defined cost function $J(w, b)$ and reach its local minimum by changing the values of its parameters: The weight parameter w and the bias parameter b . The gradient descent method is a well-known convex function. The values of parameters w and b should be optimized to obtain the minimum cost function. First, their values are initialized randomly. Then, the gradient descent algorithm starts at this point and proceeds from top to bottom as shown in Figure 2.7 until the cost function becomes as small as possible. The gradient is multiplied by a factor α , called the learning rate, and then subtracted from the previous value of the weight parameter, as shown in Eq. (2.23). At each iteration of the algorithm, the value of the loss is calculated. This procedure is repeated until convergence is achieved.

$$w(t+1) = w(t) - \alpha \left(\frac{\partial loss}{\partial w} \right). \quad (2.23)$$

It is worth noting that the learning rate α defines the steps needed to find the target (i.e., the local minima) along the slope. It also indicates how fast or slow our algorithm will proceed to obtain the optimal weights. The learning rate should be calibrated so that it is neither extremely low nor too high to reach the local minimum. If the learning rate is chosen too low, the gradient descent may reach the local minimum, but the calculation will take a very long time. However, if the learning rate is very high,

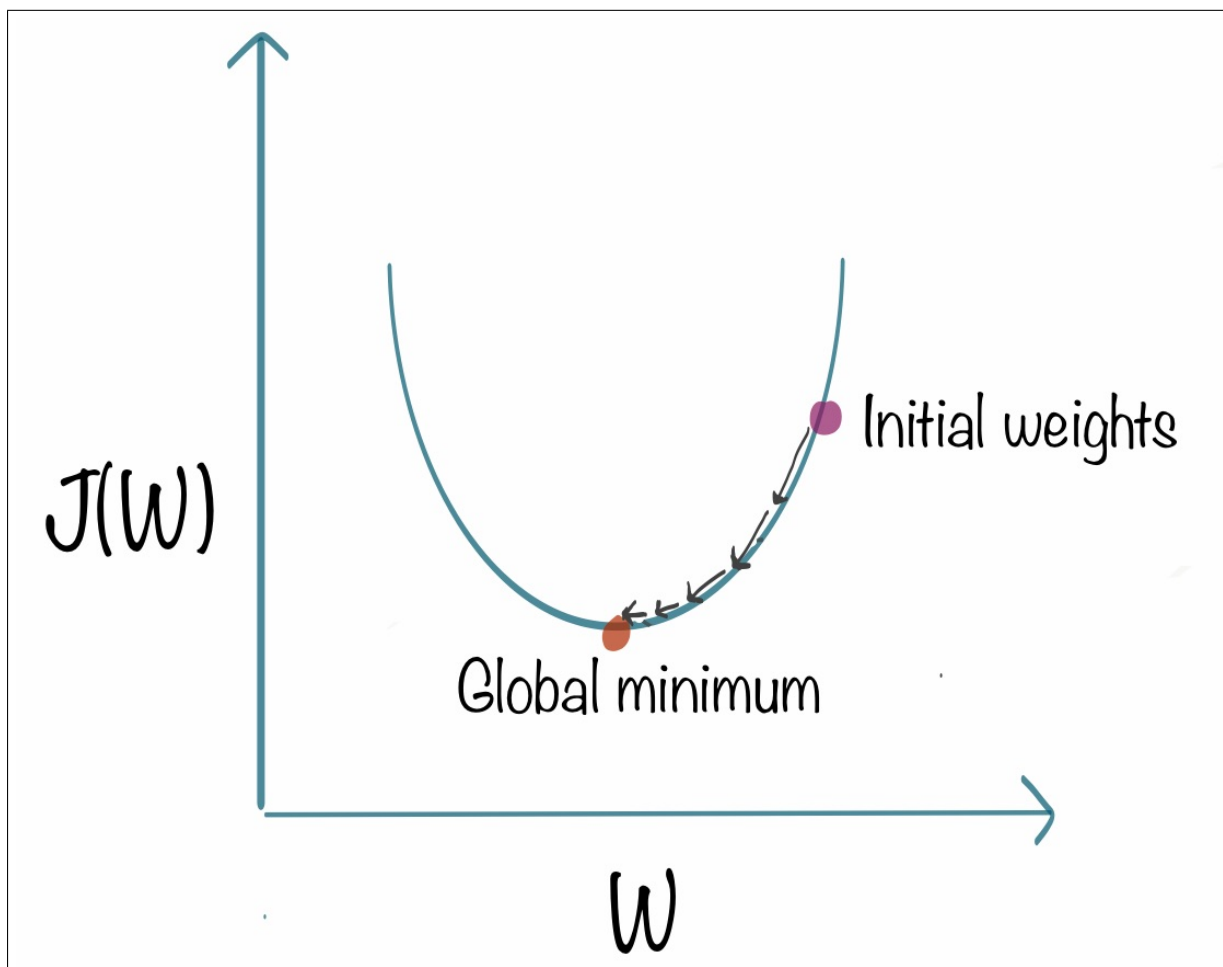


Figure 2.7: The iterative procedure of the gradient descent method [3].

the local minimum will not be reached, as shown in Figure 2.8. This problem is called the overshooting problem. Consequently, the learning rate must never be either high or low.

2.7 Overview of the Auto-weighted strategies

All our novel methods use the scheme of automatic weighting in computing the weights of each view. First, a weight parameter is assigned to each view so that the best, most representative and informative view is given a higher weight and other views have a lower weight value. Thus, the importance of each view and its contribution to the clustering task is illustrated by this weight parameter. However, most of the existing methods use an explicit weight parameter that adds an additional hyperparameter to their objective function. This leads to a high computational cost. To overcome this drawback, all of our proposed methods use an automatic computation of the weight parameter in their objective function. Therefore, the weight of each view is automatically computed during the iterations without using any additional or hyperparameters, which reduces the computation time of the algorithm.

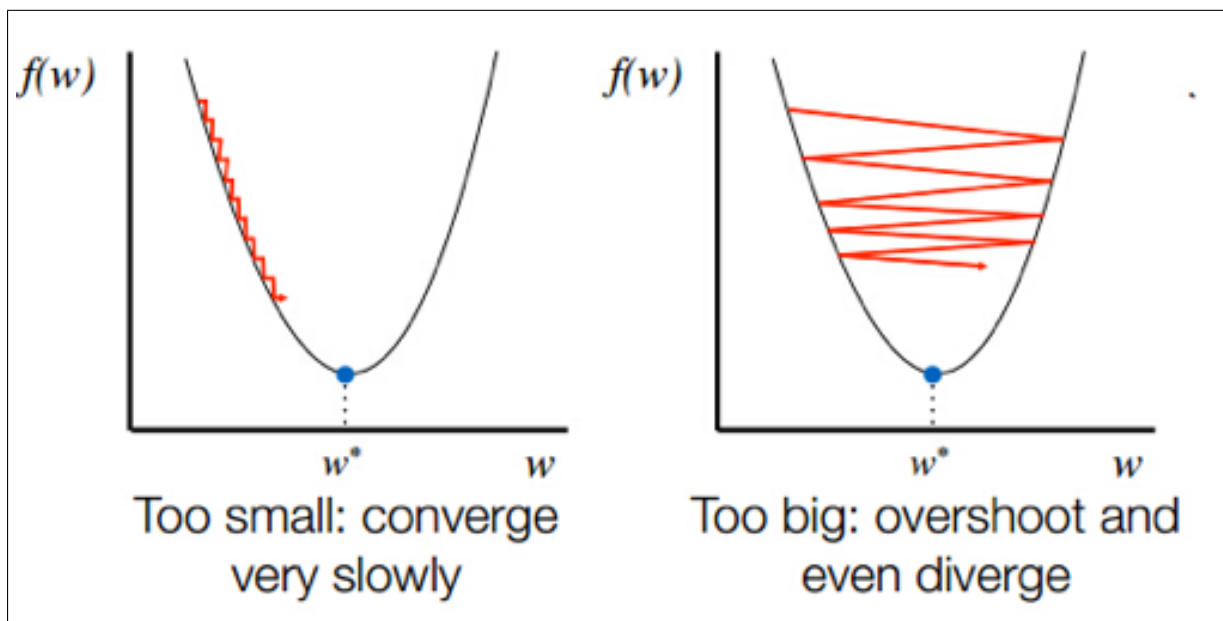


Figure 2.8: The impact of the learning rate parameter [4].

In other words: In our criterion, the weights of the data views are adaptively set and updated, taking into account the contribution of each view to the loss terms. In this way, they can reduce the impact of noisy views. Considering that the weight of each view is inversely proportional to the square root of the loss was first introduced in [13]. Therefore, this iterative trick has been used in many works aimed at minimizing an objective function consisting of an additive aggregation of several losses or terms.

2.8 Overview of the Cluster Evaluation Metrics

One of the most important aspects of clustering is the evaluation of the results without further information. Two different types of evaluation metrics can be used for clustering evaluation: 1) External criteria: They are used to determine how well the obtained cluster labels externally match the predefined class labels. 2) Internal criteria: Assess the quality of the clustering result based only on information internal to the data, without regard to external information (e.g., the silhouette coefficient). If we consider these two different categories of clustering evaluation indicators to assess the obtained clustering result, one solution would be to use external validation indicators that require prior knowledge of the ground truth classes of the data, but it is also difficult to use these indicators in real-world problems because prior information about the data set is rarely available in real-world problems. Internal evaluation indicators are a good solution in these situations. There are several external and internal indicators, each serving a specific purpose. Cluster evaluation indicators usually define the compactness of the structure of the obtained clusters.

- **Compactness:** This evaluates how similar and close the elements of a given cluster are. Variance is a typical metric for compactness.
- **Separability:** This refers to how dissimilar two separate clusters are. It determines the distance between the two clusters.

Since we have access to data sets with their labels, we used the external clustering evaluation metrics throughout our thesis. To evaluate the performance of our method, we use the four commonly known cluster evaluation metrics provided in [14], namely the clustering accuracy (ACC), the Normalized Mutual Information (NMI), the purity indicator, and the Adjusted Rand Index (ARI). These indicators can quantify the degree of agreement between the estimated clusters and the ground-truth clusters. Note that for these indicators, the higher the value, the better the result, indicating that the obtained clusters are close to the ground truth labels.

- **Purity:** The purity indicator is an external factor for cluster performance analysis. It indicates the percentage of correct labels. The calculation of purity is quite simple. Each cluster is given a label that depends on the most common class within the cluster. Purity is then calculated by dividing the number of successfully matched cluster labels and classes by the total number of observations. It is given in Eq. (2.24).

$$Purity(w_i, c_i) = \frac{1}{N} \sum_{i=1}^k \max_j |w_i \cap c_j|, \quad (2.24)$$

where N is the total number of samples, k is the number of clusters, w_i is a cluster label in W , and c_j is the ground truth label with the highest number for cluster w_i .

To better understand the concept of the purity indicator, an example is given below. Consider an example where our algorithm classifies the data set into three disjoint clusters. Each cluster in Figure 2.9 is assigned to the most frequently occurring class in the cluster. The total number of correct class labels in each cluster is calculated and then divided by the total number of samples. The purity of the clusters obtained from Figure 2.9 will be equal to: $\frac{(4+5+4)}{18} = 0.722$.

Moreover, the purity indicator increases when the number of clusters increases. This is a drawback of this indicator when the model classifies each data point in a distinct cluster, for example, the purity will become equal to one, but the model will be non performant.

Consequently, purity sometimes cannot be used as a tradeoff between the number of clusters and the efficiency of clustering.

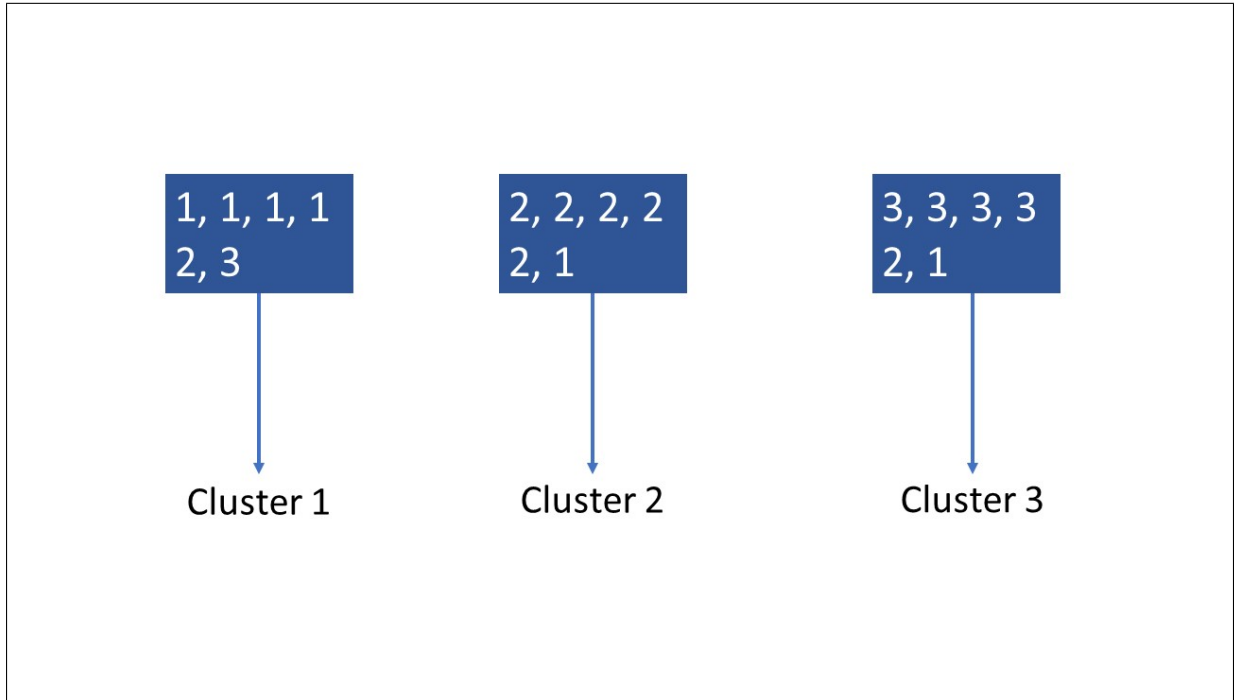


Figure 2.9: Clustering example.

- Normalized Mutual Information (NMI): The Normalized Mutual Information and information theory are closely related. First, the definition of entropy is briefly explained. Entropy is the degree of uncertainty that exists in a process. It is given by Eq. (2.25).

$$H(p) = - \sum_i p_i \log_2(p_i), \quad (2.25)$$

where p_i is the probability of the cluster label i . The entropy of the clustering obtained in Figure 2.9, is computed below. The probability of the cluster label 1 is equal to $\frac{\text{Number of samples with label=1}}{\text{Total number of samples}} = \frac{6}{18}$.

The entropy will be equal to: $(-\frac{6}{18} \log(\frac{6}{18})) - (\frac{7}{18} \log(\frac{7}{18})) - (\frac{5}{18} \log(\frac{5}{18})) = 1.089$.

Since the labels are equitably spread among the different groups, the entropy is large. Another metric used to describe the NMI indicator is Mutual Information (MI). This metric, given in Eq. (2.26) is a measure of the similarity between the cluster labels and the class labels or ground truth.

$$MI(W, C) = \sum_{i=1}^{|W|} \sum_{j=1}^{|C|} \frac{|w_i \cap c_j|}{N} \log \frac{N(|w_i \cap c_j|)}{|w_i| |c_j|} = H(W) - H(W|C), \quad (2.26)$$

where $H(W|C) = -P(C) * \sum_i P(W_i|C) \log_2(P(W_i|C))$ is the entropy of the class labels within each cluster. The MI assesses the extent to which our knowledge of class labels grows after we have been informed about cluster labels. The NMI is a normalization of the MI, with its values varying between 0 (there is no mutual information) and 1 (there is perfect correlation).

The NMI gives us an idea of the reduction in the value of the entropy of the class labels (i.e., the reduction in the uncertainty of the class labels) when the cluster labels are known. It is given by Eq. (2.27).

$$NMI(w_i, c_i) = \frac{MI(w_i, c_i)}{\frac{(H(w_i) + H(c_i))}{2}}. \quad (2.27)$$

- **Accuracy (ACC):** The accuracy indicator is the unsupervised version of the accuracy indicator used in classification tasks. It is a statistical measure that indicates the percentage of correct results out of the total number of cases examined. This indicator shows the extent to which the algorithm can correctly identify or exclude a condition. The accuracy indicator used in clustering differs from the accuracy indicator used in classification in that it uses a mapping function to determine the best mapping between the algorithm's clustering w_i and the ground truth or class labels c_i . Since the clustering method could generate a new label that is different from the actual ground truth label to describe the same cluster, the use of this mapping function is essential. ACC is given by Eq. (2.28).

$$ACC(w_i, c_i) = \max_{map \in m} \sum_{i=1}^N \frac{\mathbf{1}(map(w_i), c_i)}{N}, \quad (2.28)$$

where m is all possible permutations in $[1 : K]$, $\mathbf{1}$ is given by Eq. (2.29).

$$\mathbf{1}(map(W), C) = \begin{cases} 1, & \text{if } map(W) = C \\ 0, & \text{otherwise.} \end{cases} \quad (2.29)$$

- **Adjusted Rand Index (ARI):** The Adjusted Rand Index calculates the proportion of correct predictions among all possible predictions. It is an approximation to a measure of similarity between two different clusters by counting all pairs of observations and identifying the pairs of samples associated with the same or different clusters in the predicted and true clusters. In other words, ARI is the division between the number of pairs of data points that are either in the same cluster or in different clusters in both partitions and the total number of pairs of data points.

First, we should form a group of pairs of data points that are not in any particular sequence. For example, we consider a group of six distinct data points. The number of unique pairs formed by these data points is equal to 15. These pairs are often referred to as binomial coefficients. Consider the example in Table 2.1.

Data point	Actual label	Predicted label
A	Blue	Blue
B	Blue	Blue
C	Blue	Red
D	Red	Red
E	Red	Green
F	Red	Green

Table 2.1: Example of ground truth and the obtained cluster labels.

The total pairs of data points that can be created using the 6 data sets are: {A,B}, {A,C}, {A,D}, {A,E}, {A,F}, {B,C}, {B,D}, {B,E}, {B,F}, {C,D}, {C,E}, {C,F}, {D,E}, {D,F}, {E,F}.

ARI is given by Eq. (2.30).

$$ARI = \frac{a + b}{\binom{N}{2}}. \quad (2.30)$$

Two computed values, a and b , are important for calculating the adjusted rand index:

1. a = The number of pairs of items that are present in the same cluster for both predicted and ground truth labels.
2. b = The number of pairs of items that are in different clusters for both predicted and ground truth labels.

In the mentioned example, the elements of the pairs {A, B} and {E, F} are in the same cluster for both prediction and ground truth, so the value of a is equal to 2.

The elements in the pairs {A, D}, {A,E}, {A,F}, {B,D}, {B,E}, {B,F}, {C,E} and {C,F} are in different clusters for both prediction and ground truth clustering. Therefore the value of b is equal to 8.

Moreover, $\binom{N}{2}$ is the total number of pairs that can be generated from our data set, also known as the binomial coefficients. In our example, its value is equal to 15. Therefore, $ARI = \frac{10}{15} = 0.67$.

2.9 Overview of Deep Learning

First of all, an Artificial Intelligence (AI) is a computer system that can do things that would normally require human intelligence. Machine Learning (ML) and Deep Learning (DL) are used to drive these computer systems. The difference between AI, ML, and DL is summarized in Figure 2.10. Deep Learning has recently attracted much attention in a variety of fields. These techniques have led to significant improvements in several areas, including computer vision and natural language processing. Deep networks are used to determine complicated patterns in data that are difficult for human recognition to detect and process. Deep learning is a concept that allows

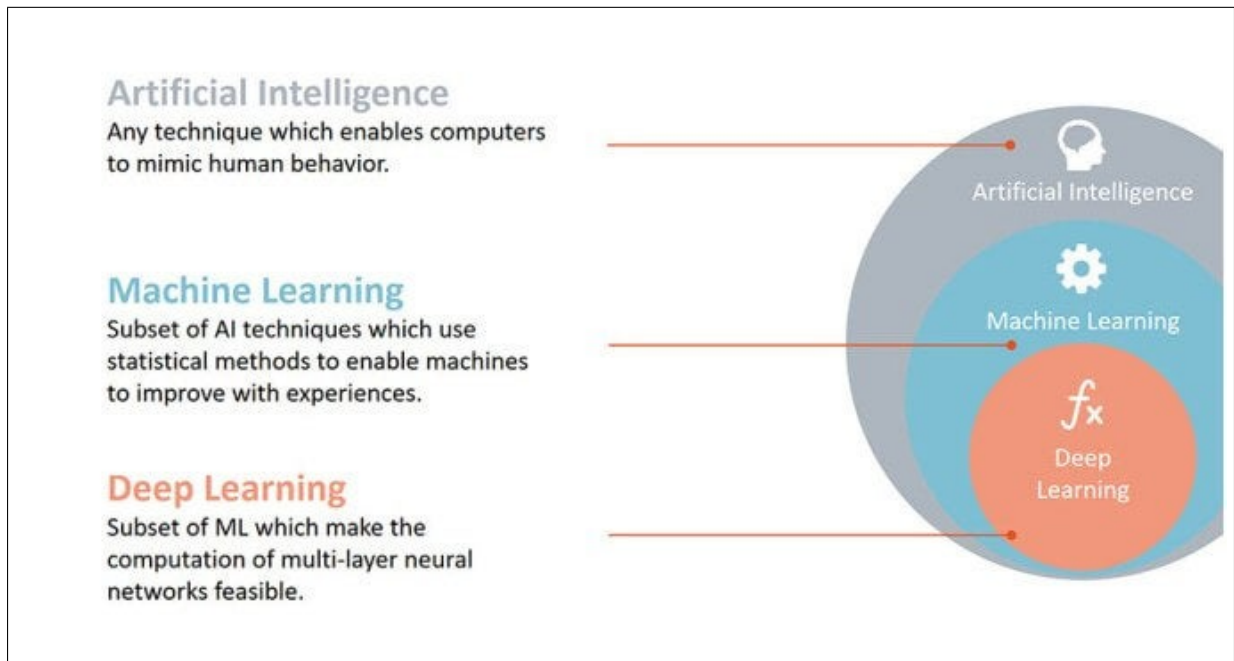


Figure 2.10: Difference between AI, ML and DL [5].

software to learn from past examples. As a result, the machine gains amazing logical capabilities through its self-learning. This is a significant step forward in the field of artificial intelligence. The computer system, which depends on a network of artificial neurons, can receive and evaluate input, decode it as information, and analyze it in relation to previously recorded information. The scope of application is huge, allowing artificial intelligence to reach its full growth potential in this field.

2.9.1 Deep Learning work

First, a brief description of the neural network is given. Neural networks, most commonly used for deep learning algorithms, analyze training samples by mimicking the interconnection of the human brain through connected layers. Each node contains input and output values, weights, and a bias (or threshold). When the calculated output value reaches a predetermined threshold, the node is activated and data is sent to the next layer of the network.

The number of hidden layers of a defined artificial neural network in a deep learning model can range from 10 to a hundred, indicating the sophistication of the model. The more layers there are, the more difficult the process becomes.

This allows a computer or model to recognise characters, words, and even entire texts. Thanks to this Deep Learning model, machines can identify a profile on a photo and even recognise and distinguish between different animals.

To recognize a cat in an image, the algorithm received the image as input and then applied the following details:

- Each different component of the image is analyzed by a particular layer of the neural network.
- At each level of manipulation of the data, the model considers one possible response; if it is "wrong", the software returns to a lower level again until a "correct" response is found. Once this is found, the next layer of the artificial neural network takes over and analyzes the potential responses.
- Once the algorithm has restructured all the information and classified the image as a cat, it can automatically classify the cat in new images.

By using Deep Learning, no other programs or codes are needed for the machine to distinguish between different types of animals. The algorithm will be able to identify the class of each image based on the training phase using the raw data provided (in this case, several photos of animals). Therefore, the same model can be applied to different types of applications and data due to its high adaptability to new situations.

The amount of raw data fed into the algorithm is critical. The more input data is provided to the model, the faster and more efficiently it will learn.

2.9.2 The main difference between Deep Learning and Machine Learning

Machine learning is a term based on the assumption of automatic learning by a machine. The result is that the computer adapts to previous data inputs and provides an appropriate solution to a difficult situation.

Machine learning has a subset called Deep Learning. Deep Learning, on the other hand, allows for independent analysis of original data, while machine learning requires prior data processing to allow a model to classify incoming data. As a result, deep learning opens up a range of new options and enables more human-like learning.

2.9.3 Convolutional neural networks

Convolutional neural networks (CNN/ConvNets) are a popular class of deep neural networks. In deep learning, a convolutional neural network is a type of artificial neural network (ANN) with different types of layers used to evaluate visual images. It is also used to classify different objects

into different classes. Convolutional operation is the basis of all CNNs. The typical structure of a normal CNN consists of four layers: 1) the input layer, 2) the convolution layer, 3) the pooling layer, and 4) the fully connected layer.

Convolution layer:

Convolution layers are the basic components of CNN. Input data points, including an image and filters, are often found in this layer, and the output vectors are a feature map. After the convolution operation is applied by the convolution layer, the image is converted into a feature map, also called an activation map. Convolution means that two matrices are convolved or combined (by applying a filter, also known as a kernel or feature detector to the first matrix), to create a new matrix. An example given in Figure 2.11 shows an input matrix convolved with a filter of dimension (3×3) . Convolution is achieved by moving the kernel over the input image by one pixel, performing matrix multiplication element by element. Each element of the image is then converted into a weighted combination of itself and its neighbors. The result is thus noted in the feature map.

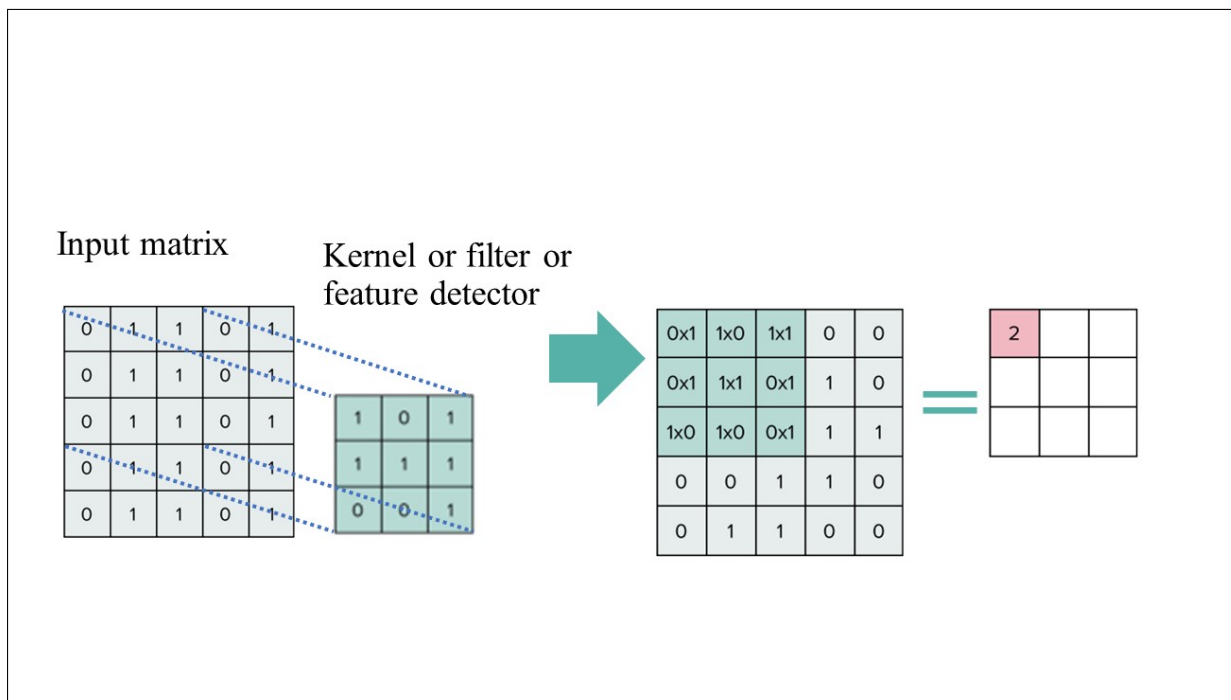


Figure 2.11: An example of the convolution operation.

The way the convolution technique is performed is affected by padding and stride. The size (height and width) of the input and output vectors can be increased or decreased by padding and stride. When the kernel or filter is applied to the image, padding specifies how many "0" fake pixels should be added to the image to take advantage of pixels at the edges. The stride of the filter, on the other hand, specifies how it convolves across the input matrix, i.e., the number of

pixels shifted. For an example of the Padding and Stride operations, see Figure 2.12.

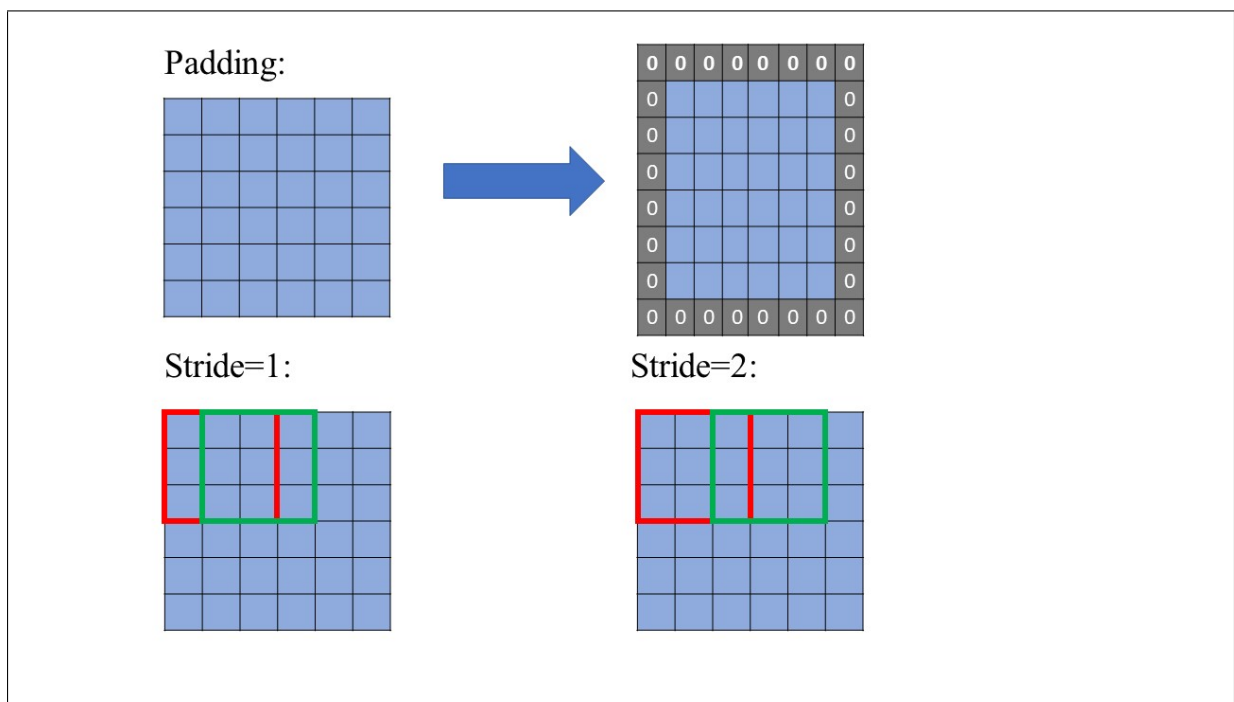


Figure 2.12: An example of the Padding and Stride operations.

Pooling layer:

To minimize the spatial complexity of the input image, the pooling layer performs downsampling. This reduces the complexity of the network and the training time, as well as the risk of overfitting. Max-pooling, where the highest value of each region in the feature map is chosen, and average-pooling, where the average of each region in the feature map is taken, are the most commonly used types of pooling. For an example of the pooling process, see Figure 2.13.

Fully connected layer:

A fully linked layer is the last layer in a CNN. All nodes from the previous layers of the network are connected together to form the fully connected layer, which is used in classification tasks. For an example of the architecture of a Convolutional Neural Network, see Figure (2.14).

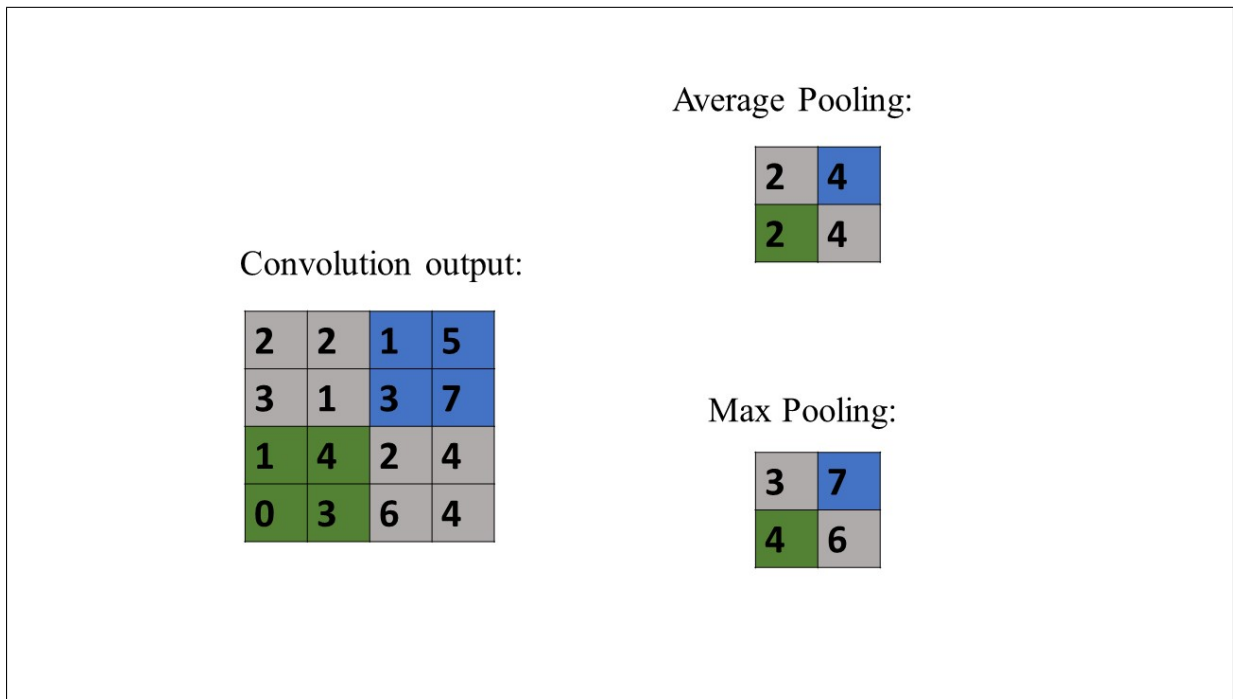


Figure 2.13: An example of the Pooling operation.

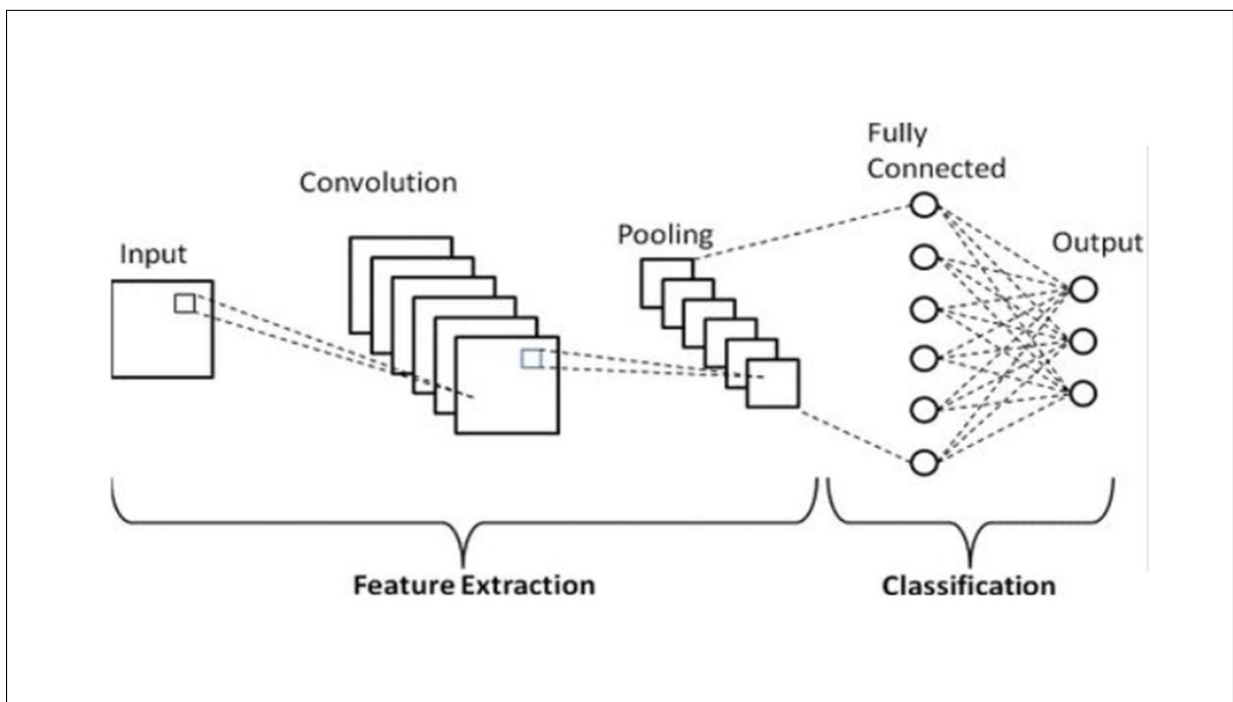


Figure 2.14: Simple Convolutional Neural Network architecture [6].

2.9.4 Definition of some descriptors

- Local Binary Patterns (LBP): The LBP is a form of visual feature used in computer vision for classification. LBP is a subset of the texture spectrum approach introduced in 1990. It

has been discovered to be an effective feature for categorising textures. The purpose of LBP is to find corners, edges, flat areas, and hard shapes in an image so that we can create a feature vector that represents the image or a collection of images. It also classifies the pixels in an image by thresholding the neighbourhood of each pixel and treating the output as binary. The extracted LBP feature is used in various applications due to its accuracy and computational efficiency, enabling image analysis in difficult real-time scenarios. For an example showing how the LBP value of a particular pixel is calculated, see Figure 2.15.

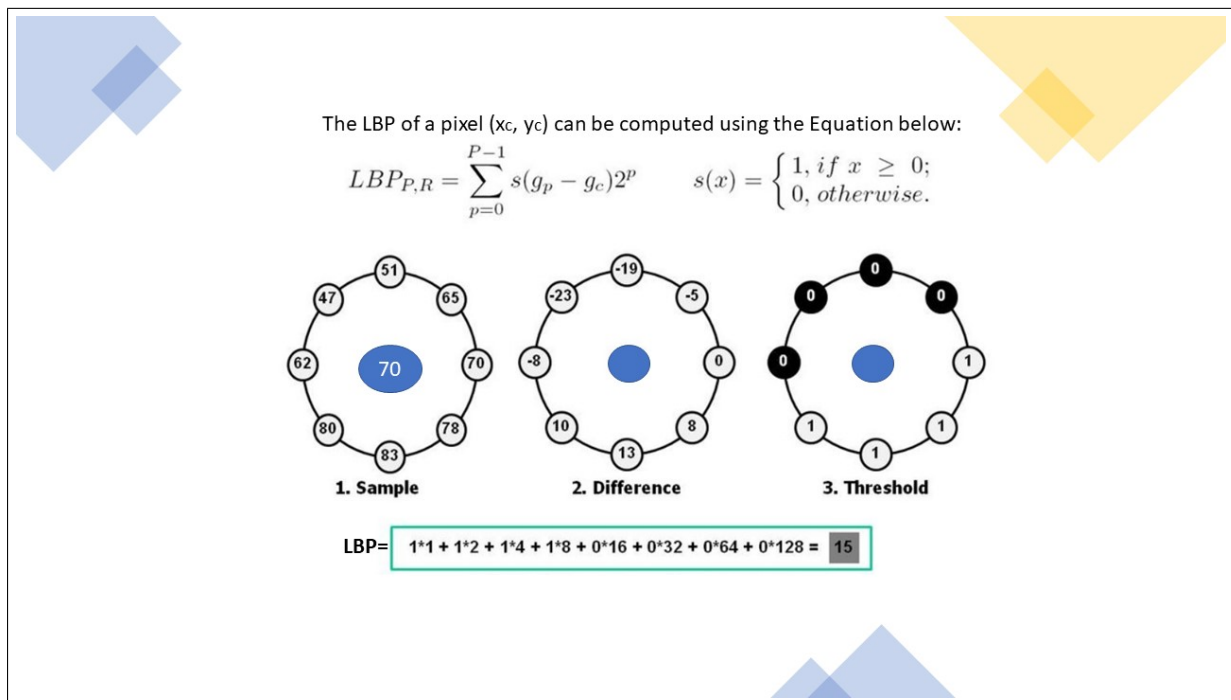


Figure 2.15: An example of the Local binary patterns feature computation [7].

- Scale-invariant feature transform (SIFT): This is an approach to feature extraction that transforms image information into local feature values that appear to be invariant to scaling, rotation, translation, and other image changes.
- Histogram of Oriented Gradients (HOG): HOG is a technique for extracting features from an image data set. It is commonly used for object recognition problems in computer vision. The HOG descriptor deals with the shape or form of an image. This extracted feature differs from edge features extracted from photographs in that the latter only determine whether a pixel appears to be an edge or not. However, the HOG feature is also able to determine the gradient and the direction of the edge. This direction is also determined in the "localized" parts of the image. This means that the overall image is divided into smaller sections, with the gradient and direction determined for each part. Then, the HOG function would create

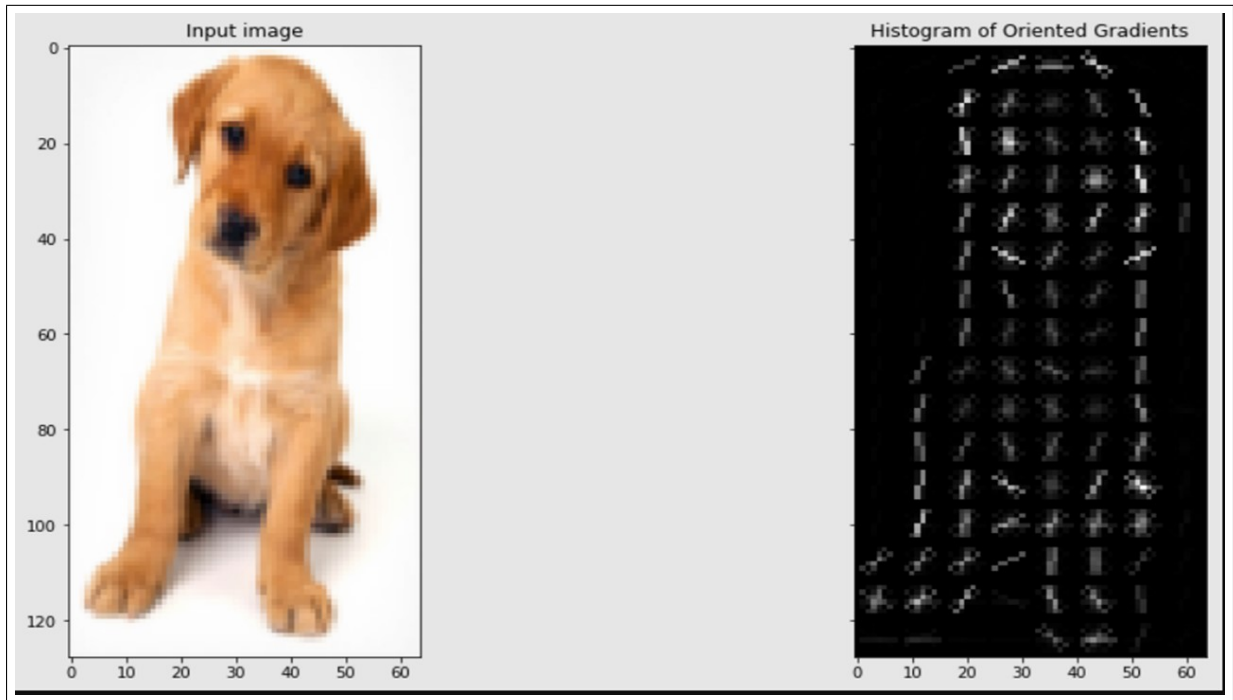


Figure 2.16: An example of the HOG feature extraction [8].

a separate histogram for those parts. The "Histogram of Oriented Gradients" is named since the histograms are created using the gradients and directions of the pixel values. For an example of the HOG feature extracted from an image of DOG, see Figure 2.16.

- Gist features: These features are named in accordance with the global image features that help characterize various significant statistics of a scene. By convolving the filter with an image at different scales and directions, these features can be obtained.
- Residual Neural Networks (ResNet50 and ResNet101): ResNets are deep descriptors used to describe each image and are based on residual connections in the network. These connections are used to solve the vanishing gradient problem of Deep Convolutional Neural Networks (DCNN), i.e., with ResNet, the error rate decreases as the number of layers increases. The concept of the ResNet descriptor is illustrated in Figure 2.17. According to this figure, element-wise addition is used. This network can be viewed as a method with a state that passes from one ResNet module to another. The output of each layer is given by:

$$x_t = H_t(x_{t-1}) + x_{t-1}, \quad (2.31)$$

where H_t is a composition of operations such as convolutional operations or pooling layers, a batch normalization, and an activation function. The activation function allows us to obtain a nonlinear boundary in the network by checking whether the neuron should be

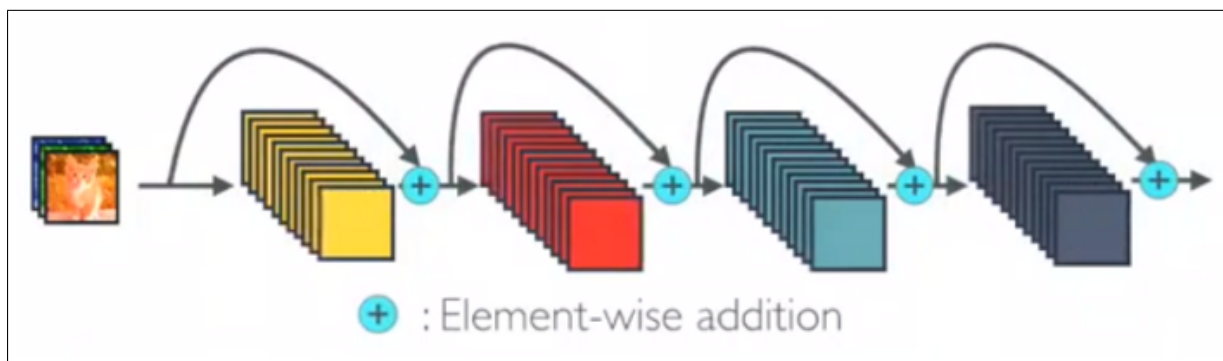


Figure 2.17: Visualization of the concept of the ResNet descriptor [9].

activated or not if the input satisfies a certain condition. If the neuron is activated, the information is transferred to the next layer. There are many activation functions that can be used for a neural network, such as Rectified Linear Unit (ReLU) and sigmoid activation functions, etc.

ResNet sums the transformed output feature and the input feature of the previous layer to get the value of the input of the next layer.

- Dense convolutional Network (DenseNet): In DenseNet, each layer is connected to all forward layers. A main advantage of this network is that it reduces the number of parameters by adding only a small set of feature maps to the other layers of the network. Based on the architecture of DenseNet, it is clear from Figure 2.18 that in DenseNet each layer is connected to all other layers. Unlike ResNets, DenseNets concatenate features and do not sum them. The output of each layer is given by:

$$x_t = H_t([x_0, x_1, \dots, x_{t-1}]). \quad (2.32)$$

- Visual Geometry Group from Oxford (VGG16): VGG-16 is a deep convolutional neural network with 16 layers. The "16" in VGG-16 means that it includes 16 layers with different weights. It is considered one of the best architectures for image processing models developed so far. One of the most striking features of VGG-16 is that instead of using a large number of hyperparameters, they focused on the assumption of a (3×3) filter convolution layer with a stride 1, and also used the same padding and max pool layer of (2×2) filter with stride 2. Throughout the scheme, the convolution layer and the max pool layer are arranged in the same way. The architecture has two fully connected layers at the end, preceded by a softmax function for output.

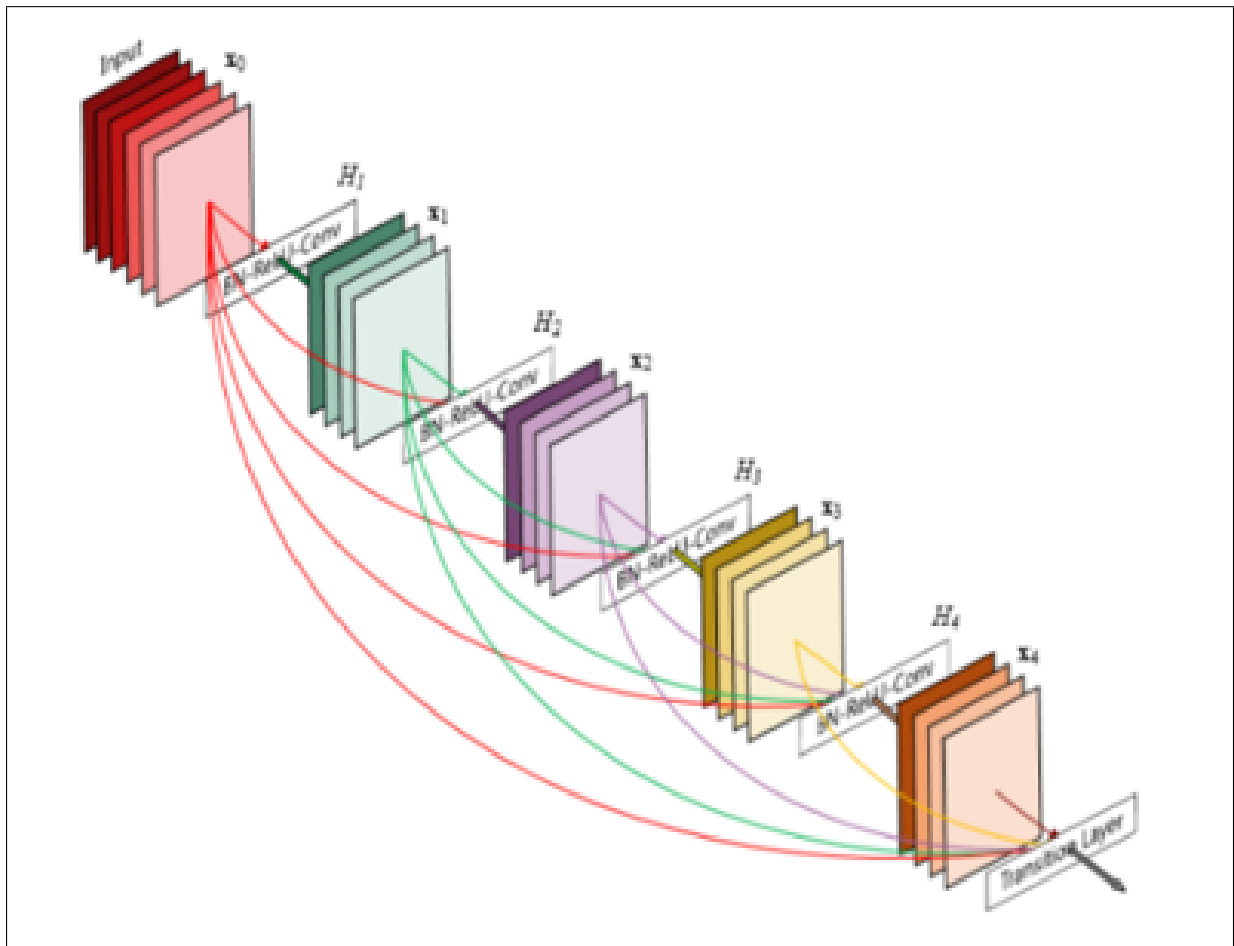


Figure 2.18: Visualization of the concept of the DenseNet descriptor [9].

Related work

Developing unique unsupervised multi-view learning approaches for clustering applications is the focus of our thesis. Several multi-view clustering methods have been developed to improve dataset processing. The obtained results show that our proposed methods outperform the state-of-the-art methods. In this chapter, the related works to our methods are described in detail.

3.1 Related Work

Recently, several multi-view clustering approaches have been proposed. The current approaches can be classified into several groups: **Spectral clustering algorithms** [12], **Graph-based clustering algorithms** [15], **Weighted multi-view clustering approaches** [16, 17, 18, 19, 20, 21, 22], **Automatically weighted multi-view clustering algorithms** [23, 24, 25, 26, 13], **Multi-view subspace based clustering approaches** [27, 28], **Kernel based Approaches** [24, 29, 10], **Matrix factorization approaches** [30, 31, 11], **Nonnegative matrix factorization methods** [32, 11], etc. In this section, we present several methods belonging to these categories.

Recently, the spectral clustering approach (e.g., [33, 34, 35, 20, 36, 37]) has attracted much attention in the field of multi-view clustering due to its simple implementation. The principle of spectral clustering is based on using the top eigenvectors of the normalized Laplacian matrix, derived from the similarity matrix between data samples. The eigenvectors of the graph Laplacian are used as a nonlinear projection of the raw data. Second, each row of the matrix formed from the eigenvectors of the Laplacian matrix is treated as a data point. Then an arbitrary clustering algorithm is used to cluster the instances in the obtained nonlinear space. This algorithm can be a post-processing step such as k-means clustering. Several works are based on the principle of spectral clustering.

An example of a spectral clustering algorithm is the co-training approach in [34], which is based on the assumption that the data point should be assigned to the same cluster in different views. To do this, spectral clustering is solved for each view to obtain the corresponding eigenvectors. Then, the eigenvectors of the first view are used to cluster the points in that view, and this clustering result is used to modify the graph of the other view. In addition, this method preserves Laplacian matrices with low rank, which is especially advantageous in clustering with multiple views at large scale, since large matrices can increase the complexity of the clustering process and take a lot of computation time. Moreover, this method does not need to specify hyperparameters, which is very encouraging in unsupervised learning. However, in most cases, the underlying clustering result is assumed to be the same for all views, which reinforces the idea of compatibility between different views. Another popular approach for spectral clustering is the co-regularized spectral clustering approach presented in [38], which adaptively combines multiple graphs from different views to improve clustering performance. For simplicity, the authors use a common regularization parameter for all pairwise co-regulators in their objective function. However, it would be better if different values of the parameter are used for different pairs of views to account for the best contribution of each view. Moreover, a k-means step is used to obtain the clustering result, which strongly depends on the initialization. Furthermore, numerous spectral clustering algorithms have been used for multi-view clustering [39, 40, 41]. These methods construct a graph for each view that represents the similarity between data points. The nodes of the graph represent the samples, and the edges represent the similarity between these samples. Then, the spectral projection matrix of each similarity matrix is constructed and a post-processing step is applied to obtain the clustering assignment.

Weighted multi-view clustering approaches, such as the methods proposed in [16, 18], are another multi-view clustering approach in which the authors add a weight for each view such that the best view has the highest weight and thus contributes the most to the clustering task. However, these methods require additional parameters to assign a weight to each view.

To overcome these drawbacks, auto-weighted multi-view clustering algorithms were proposed in [23, 42, 43]. These algorithms eliminate the complexity of using additional parameters by automatically updating the weights of each view. Moreover, a Multi-view Learning with Adaptive Neighbors algorithm (MLAN) is proposed in [44] to jointly estimate the graph matrix and the clustering task. The main advantage of this method is that there is no explicit weighting parameter for each view in its objective function. It has the ability to learn the weighting parameters automatically. It is almost parameter free, which makes it easier to deal with real world

situations. Moreover, the proposed method allows clustering multiple views and learning local structures simultaneously. The approach imposes a constraint on the similarity matrix to contain c connected components to increase accuracy. However, this method consists of matrix inversion and eigenvalue decomposition steps, which are time-consuming. In addition, feature selection methods have attracted much attention to selecting the best and most informative features that can reduce the effects of noise. For example, in [16], the authors developed a new method that uses two weighting schemes. The first is to model the importance of each view, and the second is to select the best features of each view. These two steps are computed simultaneously with the k-means post step to obtain clustering results that lead to better performance.

Another approach related to the above categories is presented, which extends the spectral clustering algorithm with the idea of weighted views. This method is called [20]. This method is called "Adaptively Weighted Procrustes" (AWP) and is a version of spectral-based clustering that uses spectral rotation to learn the cluster indicator matrix. This approach features low computational cost and high precision compared to other graph-based methods.

Another famous category called Multi-view Subspace based Clustering approach (MVSC) was introduced in [45, 46, 47, 48, 49, 50]. These algorithms project the data into different subspaces to learn the most consistent representation of the data so that it can be clustered correctly. In other words, these methods learn a unified representation from multiple subspaces of all views of the data or learn a latent space and then return the unified representation. In addition, a new method called Smoothed Multi-View Subspace Clustering (SMVSC) is proposed in [51]. This approach uses a new technique called graph filtering to create a smoothed representation for each view with similar feature values of similar samples. This is done using low-pass filtering to preserve the geometric features of the graph. In this way, a "cluster-friendly" structure is created that greatly simplifies the subsequent clustering process. In [52], the authors use the latent representation of the datasets to group them into different clusters and jointly analyze the underlying complementary information from different views. Unlike other previous approaches for single view subspace clustering, where the different samples are grouped based on original features, this method learns the consistent similarity matrix in the latent space, which leads to better results than using the raw data to build the graph. The work of [27] introduced a subspace clustering algorithm that constructs the most consistent graph similarity matrix with a large spectral gap based on the joint shadow p-norm and spectral clustering. Another example of multi-view subspace clustering methods is the method presented in [53]. In this method, the affinity matrices of the different views are merged at the partition level rather than directly,

reducing the effects of noise and outliers in the original data. In addition, this method jointly estimates the affinity matrix of each view, the consensus representation, and the final cluster assignment. Performing these three steps jointly can improve the performance compared to other methods.

The kernel-based approaches (e.g., [24, 54]) are used to overcome the problem of nonlinearity of the data by mapping them to a space in which they are linearly separable, and then they solve the problem caused by the multiple shapes of the datasets. In [24], the authors present two auto-weighted multi-view clustering approaches that exploit kernelized graph learning. The first approach uses a single kernel to map the data into a space where they are linearly separable. The second approach is characterised by its ability to use a combination of multiple kernel matrices, where the performance of the method often depends on the input kernel matrix. Using these two methods, it is possible to simultaneously estimate a consistent similarity graph, a consistent spectral projection matrix, and the weights of each view without additional parameters. In [55], the authors propose a method that learns a unified graph by simultaneously estimating the self-expressing coefficients and the affinity matrix from multiple kernels. They perform the final clustering on the obtained graph. The work of [56] is an automatically weighted multi-kernel approach, which can solve the problem caused by non-linear data and noise by using the mixture correntropy to measure the similarity between each kernel and the common kernel matrix.

Moreover, matrix factorization approaches (e.g., [30, 31, 11]) are used due to their low computational cost, which makes them efficient for dimensionality reduction. They can provide high clustering performance compared to other methods. For example, the method in [30] called Integration by Matrix Factorization (IMF) generates different representative clustering matrices computed independently for each view, in addition to an intermediate matrix for all views, and then performs a factorization process on this matrix to reconcile the different clustering matrices generated from the different views. Matrix factorization methods have a low computational cost compared to other methods. This is because the factorization of a given matrix decomposes it into its constituent parts, which can also simplify the matrix operations since they are applied to the obtained matrices and not to the original complex matrix. However, these methods cannot handle the nonlinearity of the data.

Furthermore, a Nonnegative Embedding and Spectral Embedding (NESE) method is proposed in [11]. The main idea behind [11] is to directly estimate the nonnegative embedding matrix \mathbf{H} (cluster label matrix) from the individual graph matrices \mathbf{S}_v and the individual spectral representations \mathbf{S}_v using $\mathbf{S}_v \approx \mathbf{H}\mathbf{P}_v^T, v = 1, \dots, V$. It provides the clustering result directly without

any additional parameters or post-processing steps. In addition, this method simultaneously provides the consensus nonnegative embedding matrix and the spectral representation matrices. In [32], the authors propose a Nonnegative Matrix Factorization (NMF) approach that uses dual constraints. This approach exploits the labeling of some images and the sparsity of the representations. Nonnegative Rank-Reduced regression (NRRR) is presented in [57]. In this work, the authors exploit distance metric learning and clustering by introducing a unified framework for rank-reduced regression. The approach provided some new insights for learning a cluster partition that takes advantage of distance metric learning. In [58], a multi-view nonnegative matrix factorization is proposed. The model, which estimates the view-based two nonnegative matrices, integrates manifold regularization in the low-dimensional subspace and the pairwise consistencies of interview similarity in these low-dimensional subspaces.

In addition, many graph-based multi-view clustering methods have been developed. The first approach in [59] can jointly learn the similarity graph matrix, the unified graph matrix, and the final clustering assignment by using a novel multi-view fusion technique that automatically assigns a weight to each graph matrix to obtain the unified graph matrix. This approach imposes a rank constraint on the Laplacian matrix to obtain exactly K clusters.

Moreover, in [60], the authors propose an Ensemble Clustering by Propagating Cluster-wise Similarities with Hierarchical Consensus approach (ECPCS-HC) and an Ensemble Clustering by Propagating Cluster-wise Similarities with Meta-cluster-based Consensus method (ECPCS-MC). These two methods use random walks based on effective propagation of cluster-wise similarities via random walks. Moreover, these two methods propose two new consensus functions to obtain the final clustering result. The work of [61] proposed a Locality Adaptive Latent MultiView Clustering (LALMVC) method. It simultaneously learns the latent consensus representation via linear transformations, the joint spectral representation and the consensus graph. The learned consensus graph matrix is then used in spectral clustering to obtain a cluster index matrix. In [62], the authors propose a model where the individual graphs, a fused graph, and a spectral projection are estimated simultaneously. The self-representativeness of the data was used in estimating the individual graphs. In [63], the authors propose a framework for graph learning. Their method learns initial graphs from data instances in different views. These are then optimized using a low-rank constraint Laplacian matrix. Therefore, these estimated graphs are incorporated into a global and unified graph estimation. This unified graph integrates the same rank constraints over its Laplacian matrix. Thus, the clustering result is obtained directly from this graph without any post-processing step.

Many existing methods use two separate steps to obtain the clustering result. The first step is based on learning the joint affinity matrix, and the second step is used to obtain the clustering result by applying a hard clustering method such as k-means clustering. To eliminate the problem of inconsistency caused by the fact that the goal of the first step is not to obtain an optimal clustering performance, a new method is presented in [64]. This method, called One-step Multi-view Spectral Clustering (OMSC), integrates the steps of learning the affinity matrix of each view and the joint affinity matrix learned from the low-dimensional space of the data, as well as the step of k-means clustering into one framework. The joint affinity matrix is considered as the final clustering assignment. Moreover, the weight of each view is learned automatically to reduce the impact of noisy views. In [65], the authors jointly estimate an optimal graph and an adequate consensus kernel for clustering by forcing the global kernel matrix to be a convex combination of a set of basis kernels. Their proposed model enforces a regularization of the unified graph and the final kernel matrix.

In [66], the authors propose a new method that solves the inconsistency problem using a one-step scheme: one-step multi-view spectral clustering by learning Common and Specific Nonnegative Embedding (CSNE). This work is an improved version of the NESE method. It divides the nonnegative embedding matrix used in NESE into two matrices. The first matrix denotes the joint nonnegative embedding matrix \mathbf{H}^* , which represents the joint cluster structure, and the specific nonnegative embedding matrix \mathbf{H}_v , which represents the specific cluster structure for each view. This method is better at reducing the effects of noise and outliers than the method NESE. In [67], the authors describe a new method called Multi-View Clustering in Latent Embedding Space (MCLES). This method can learn both the latent embedding space of the data and the cluster indicator matrix simultaneously. Using the latent embedding space, it can learn the global structure of the data and the complementary information between all views. A main advantage of this method is that it can limit the effects of noise that may be present in the raw dataset. In [68], the authors use the correntropy-induced metric (CIM) to deal with the noise in each view. They use a view-specific embedding from an information theoretic perspective. In [69], the authors propose the algorithm Cross-view Matching Clustering (COMIC), which can cluster data with multiple views. The algorithm can also estimate the number of clusters. COMIC provides cross-view consensus on view-specific similarity graphs instead of view-specific data representations.

Another recent work in [70] addresses clustering of multi-view data, where some data instances are missing in some views, using spectral perturbation theory. First, the similarity matrix of each

view is constructed. The average similarity values of the other views containing these missing samples fill in the missing values in the similarity matrix. The second step of this method is learning the consensus matrix. First, the Laplacian matrix of each similarity matrix is calculated. Then, using perturbation theory, a weight is assigned to each Laplacian matrix to obtain the final consensus Laplacian matrix, which is used for the final clustering result.

In [71], the authors present a method called Multi-view Cluster Analysis with incomplete data to understand treatment effects. Indeed, data entries are sometimes missing in several of the views. Current multi-view co-clustering approaches are not able to successfully deal with incomplete data, especially when there are many patterns of incomplete data. By using an indicator matrix whose entries indicate which data items are present and measuring clustering performance based solely on observed values, this method provides an improved approach for multi-view co-clustering algorithms to deal with the missing data problem. In addition, this method is less susceptible to imputation uncertainty than standard methods that substitute missing data to perform regular clustering of multi-view data. To alleviate the problem of missing data when analyzing data from multiple sources, a completion scheme for missing data in multiple views based on regularized nonnegative matrix factorization in multiple manifolds was presented in [72]. This approach was based on the assumption of consistency of multi-view data, and a multi-manifold regularized nonnegative matrix factorization algorithm was employed to obtain a homogeneous manifold and global clustering.

Many other multi-view clustering methods are presented below.

In [73, 74], the authors present a Density-Based clustering technique, namely DBSCAN. This approach assumes that clusters are dense regions in space separated by less dense regions. It creates different clusters based on the local density of each data point. The robustness of the DBSCAN clustering algorithm to outliers is its main feature. Also, unlike the famous k-Means clustering algorithm, which requires the number of clusters to be fixed, it does not require the number of clusters to be known in advance. However, a major drawback of the DBSCAN algorithm is that it cannot cluster data points with large density differences.

Furthermore, in [75], the Density Peak Clustering (DPC) technique is presented. This method assumes that cluster centers are located in the local high-density region and groups other data points by assuming that they are in the same cluster as their nearest higher density neighbors. However, this method does not take into account the representation and structure of the data and therefore cannot accurately identify the noise nodes.

In [76], the authors present an approach called the Dual Shared-Specific Multi-view Subspace Clustering method (DSS-MS). This method simultaneously explores the features of each view in the low-dimensional space to exploit the relevant and specific information of each view in addition to the correlations between the shared information across different views, using a dual learning framework.

Also, in [77], the authors propose a new method known as Multi-View Spectral Clustering via Sparse Graph Learning (S-MVSC). This method learns a sparse and common affinity matrix for all views to reduce the effects of noise and outliers in all views. This technique has the same computational cost as the popular single-view spectral clustering method, which means that this method is faster than some multi-view methods.

In [78], the authors presented a Consistency and Inconsistency-aware Graph-based Multi-View Clustering (CI-GMVC) method that integrates the consistent and inconsistent parts across different views. This method examines the consistency and inconsistency in the individual similarity matrices and splits them into two corresponding graphs (consistent and inconsistent graphs) by using the orthogonality constraints. Therefore, a unified similarity matrix is created from the consistent parts of the similarity matrices. This matrix is used to directly give the final clustering result.

The canonical correlation analysis (CCA) [79] extracts some of the features from different views by projecting the dataset into a low-dimensional space and then exploring the correlation between different views. Thus, it applies any clustering algorithm like k-Means algorithm to learn the partition.

In [80], the authors propose a method called Fine-grained sImilariTy fuSion for Multi-view Spectral Clustering (FITS-MS). This method can overcome the problem of assigning the same weight to all samples in a view; in incomplete views, some elements may be distorted or absent, while others remain intact in all views. Moreover, sparse subspace clustering is used to construct the initial similarity matrices, which yields encouraging results. They also propose a fine-grained similarity fusion technique to obtain the final consensus affinity matrix to address the shortcomings of coarse-grained information fusion. During the procedure, the local inter-view and the global intra-view weight relationships are explored. FITS-MS is particularly convenient because it has only one hyperparameter.

Another method was developed in [81]. This method is called Multi-View clustering based on Orthonormality-Constrained Nonnegative Matrix Factorization (MVOCNMF). Based on the label

information, the proposed method first learns the low-dimensional space of the dataset using the constrained NMF method and then clusters the instances with the same label into clustering prototypes for each view. Then, the authors propose a new orthogonal constraint term to achieve the desired representations for each view and apply the co-regularisation method to combine the complementary information from different views.

In [82], the authors propose Robust Self-Tuning Multi-View Clustering (RST-MVC). In this work, the authors present a method that uses a sum-of-norm loss function to reduce the initialization sensitivity problem, a sum-of-norms regularization to automatically estimate the number of clusters, and integrates a strong statistical method to reduce the effects of outliers.

Two surveys of multi-view clustering can be found in [83, 84]. These papers summarize a large number of multi-view clustering algorithms, including generative and discriminative methods. In addition, the authors classify these algorithms into different categories and give many examples of how these algorithms are used for multi-view clustering.

Table 3.1 summarizes and presents part of the literature review with year and authors, objective, variables, and types of datasets.

Table 3.1: Some related multi-view clustering methods.

Author and year	Method	Objective	Variables	Type of datasets
[34] A. Kumar and H. Daumé (2011)	Co-training	The spectral embedding from one view is used to constrain the similarity graph used for the other view	Input: Similarity matrices of all views Output: Spectral projection of each view	Synthetic and real datasets (documents and handwritten digits)
[38] A. Kumar, P. Rai, and H. Daumé (2011)	Co-Regularized	Enforces the view pair's eigenvectors to have a strong pairwise similarity by forcing them to share a common centroid matrix	Input: Kernel matrices of all views Output: Eigenvectors matrices of view pair and the common centroid matrix	Synthetic and real datasets (documents, handwritten digits and images)

[44] F. Nie, G. Cai, J. Li, and X. Li (2017)	MLAN	Performs clustering and local structure learning simultaneously	Input: Raw data matrices of all views Output: Spectral projection matrix, and Consistent similarity matrix	Real datasets (Images and handwritten digits)
[20] F. Nie, L. Tian, and X. Li (2018)	AWP	Weights the views according to their clustering capacities, resulting in a weighted Procrustes average problem	Input: Spectral embedding matrices for all views Output: Rotation matrices of all views and cluster index matrix	Real datasets (Images and handwritten digits)
[64] X. Zhu, S. Zhang, W. He, R. Hu, C. Lei, and P. Zhu (2018)	OMSC	Learns the common affinity matrix from low-dimensional data and the clustering result in one step	Input: Raw data matrices of all views Output: Individual affinity matrices of all views, common affinity matrix, common representation of data with multiple views and low-dimensional space	Synthetic and real datasets (documents, handwritten digits and images)
[24] S. Huang, Z. Kang, I. W. Tsang, and Z. Xu (2019)	MVCSK	Simultaneously performs multi-view clustering and finds the similarity matrix in kernel spaces	Input: Kernel matrices of all views Output: Spectral projection matrix and consistent similarity matrix	Real and synthetic datasets (documents, web pages and images)
[11] Z. Hu, F. Nie, R. Wang, and X. Li (2020)	NESE	Estimates the nonnegative embedding matrix to give the final clustering result	Input: Similarity matrices of all views Output: Nonnegative embedding matrix, and spectral projection matrices of all views	Real datasets (images)

[67] M.-S. Chen, L. Huang, C.-D. Wang, and D. Huang (2020)	MCLES	Simultaneously estimates the global structure and cluster membership matrix in a learned latent embedding space	Input: Raw data matrices of all views Output: Latent embedding space, spectral projection matrix, mapping models and consistent similarity matrix	Real datasets (documents and images).
[81] H. Cai, B. Liu, Y. Xiao, and L. Lin (2020)	MVOCNMF	Learns the low-dimensional space of the dataset using the constrained NMF method, clusters the samples into the clustering prototypes, and applies the co-regularization approach to combine all views	Input: Raw data matrices of all views Output: Label constraint matrices of all views, low-dimensional representations of all views, average value of all low-dimensional representations	Real datasets (documents and images)
[77] Z. Hua, F. Niea, W. Changa, S. Haoa, R. Wang, X. Li (2020)	S-MVSC	Learns a consensus similarity matrix \mathbf{W}^* from multiple views, and performs Ncut to \mathbf{W}^* to obtain the clustering result	Input: Raw data matrices of all views, Similarity matrices of all views Output: Consensus similarity matrix \mathbf{W}^*	Real datasets (documents and images)
[66] H. Yin, W. Hu, F. Li, and J. Lou (2021)	CSNE	Estimates the common nonnegative embedding matrix and the specific nonnegative embedding matrix to give the final clustering result	Input: Individual similarity matrices of all views Output: Common nonnegative embedding, spectral embedding matrices of all views, and specific nonnegative embeddings of all views	Real datasets (documents and images)

[78] M. Horie and H. Kasai (2021)	CI-GMVC	Learns the consistent and inconsistent parts of the similarity matrix across multiple views, and create a consensus similarity matrix \mathbf{U} from the consistent parts of the similarity matrices	Input: Individual similarity matrices \mathbf{S}^v Output: Unified similarity matrix \mathbf{U}	Real datasets (documents and images)
[80] X. Yu, H. Liu, Y. Wu, and C. Zhang (2021)	FITS-MS	Generates the consensus similarity matrix using a fine-grained similarity fusion strategy, handles the problem of missing data, and explores the local and global relationships between views	Input: Raw data matrices of all views Output: Similarity matrices of all views, auxiliary variables of all views to substitute the corresponding similarity matrices, and consensus similarity matrix	Real datasets (documents, handwritten digits and images)

3.2 Typical approaches

Multi-graph fusion for multi-view spectral clustering [62]:

This method jointly estimates view-specific graphs, a unified graph, and a consistent spectral embedding. The learning model is based on the use of self-representativeness of the data and is given by:

$$\min_{\mathbf{S}^v, \mathbf{S}, \mathbf{P}} \sum_{v=1}^V \left\{ \|\mathbf{X}^{(v)} - \mathbf{X}^{(v)} \mathbf{S}^v\|_F^2 + \alpha \|\mathbf{S}^v\|_F^2 + \beta \|\mathbf{S}^v - \mathbf{S}\|_F \right\} + \gamma \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}), \quad (3.1)$$

where \mathbf{S}^v is the v -th view's graph, \mathbf{S} is the unified graph, and $\mathbf{P} \in \mathbb{R}^{n \times C}$ is the consistent spectral embedding. α , β , and γ are three balance parameters.

Graph learning for multi-view clustering (MVGL)[63]:

Using V distinct view graphs \mathbf{S}^v ($v \in \{1, \dots, V\}$), the authors estimate a unified consistent graph,

$\mathbf{A} \in \mathbb{R}^{n \times n}$, by minimizing the following criterion:

$$\begin{aligned} & \min_{\mathbf{A}, \mathbf{P}, w_j^{(v)}} \sum_{i,j=1}^n \left(A_{ij} - \sum_{v=1}^V w_j^{(v)} S_{(i,j)}^v \right)^2 + \gamma \text{Tr}(\mathbf{P}^T \mathbf{L}_a \mathbf{P}) \\ & \text{s.t. } \mathbf{A}^T \mathbf{1} = \mathbf{1}, \mathbf{A} \geq 0, \mathbf{P}^T \mathbf{P} = \mathbf{I}, \sum_{v=1}^V w_j^{(v)} = 1, \forall j, \end{aligned} \quad (3.2)$$

where \mathbf{L}_a is the Laplacian matrix of the similarity matrix $\frac{\mathbf{A} + \mathbf{A}^T}{2}$, $\mathbf{P} \in \mathbb{R}^{n \times C}$ is the unified spectral projection, and $w_j^{(v)}$ is the weight of column j in view v . There are nV unknown weights. γ is a regularization parameter.

Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding (NESE)[11]:

The authors in [11] developed a method called "Multi-view spectral clustering via integrating Nonnegative Embedding and Spectral Embedding" (NESE) [11]. Inspired by the factorization of nonnegative matrices, the authors propose in [11] a new approach that can get the clustering result directly and avoid the need for post-process clustering and using additional parameters by finding the nonnegative embedding and the spectral embedding matrix simultaneously. Inspired by the symmetric nonnegative matrix factorization and the relaxed continuous Ncut (i.e., the spectral embedding), the authors developed a new objective function in [11] to estimate a consistent nonnegative embedding matrix \mathbf{H} . The objective function of NESE is:

$$\min_{\mathbf{H}, \mathbf{P}^v} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2 \text{ s.t. } \mathbf{H} \geq 0, \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \quad (3.3)$$

where \mathbf{S}^v is the graph matrix associated with view v , \mathbf{P}^v is the corresponding spectral projection matrix, and \mathbf{H} is the consistent nonnegative embedding used to obtain the cluster results directly without additional parameters or a post-processing method such as k-means or spectral rotation, where the results may depend heavily on the initialization. The input of this approach is the similarity matrix of each view. Both the spectral projection matrix and the unified nonnegative embedding matrix are unknown and are estimated by an iterative optimization algorithm.

Auto-weighted multi-view clustering via kernelized graph learning (MVCSK)[24]:

The MVCSK method constructs a unified graph from the kernel matrices using the concept of self-representation of data in high-dimensional spaces. Therefore, this method can handle nonlinear data. It estimates the graph matrix and spectral embedding matrix by minimizing the following

objective function:

$$\min_{\mathbf{S}, \mathbf{P}} \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})} + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) \quad s.t. \quad \mathbf{S} \geq 0, \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}, \quad (3.4)$$

where μ and λ_1 are two positive regularization parameters. The first term is a sum over the views of the self-reconstruction error of the data in a high-dimensional space. This space is defined by the corresponding kernel matrix. The second term is a simple regularization that promotes graph matrices with a small ℓ_2 norm. This is often used to overcome overfitting problems. The third term specifies that the graph should be as close as possible to a graph with exactly C connected components (clusters). It is worth noting that this term is exactly what is minimized to estimate a spectral embedding once the graph is known.

Using the square root of the reconstruction error in (3.4) allows the use of automatic weighting for each view. This means that a view that has a small reconstruction error will receive a large weight. Unlike other approaches, the above formulation is able to update the weight automatically, eliminating the complexity of using additional parameters. The weight, w_v , of each view is given by:

$$w_v = \frac{1}{2 \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})}}. \quad (3.5)$$

Using the above weight, it can be shown that problem (3.4) is equivalent to the following problem:

$$\min_{\mathbf{S}, \mathbf{P}} \sum_{v=1}^V w_v \text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) \quad s.t. \quad \mathbf{S} \geq 0, \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (3.6)$$

Experimental setup and Datasets

4.1 Motivation

Considering multi-view clustering algorithms, six different methods are provided in this thesis. All of these methods have the same goal: clustering different multi-view data into distinct groups. We also provide a comprehensive analysis of all the methods discussed.

Numerous experiments with the same experimental setup (type of kernel, initialization of matrices, etc.) have been performed to demonstrate the superiority of the proposed methods compared to other state-of-the-art methods.

4.2 Datasets

In this study, we used a variety of real and synthetic datasets. The datasets tested by our methods include image datasets, documents, UCI Digits datasets, and scene datasets. Although most of our methods are based on clustering real images, we also used some synthetic datasets.

4.2.1 Description of the datasets used in our methods

- **COIL20 [85]**: The database is called the Columbia Object Image Library and consists of 1440 grayscale images. These images are grouped into 20 groups, each consisting of 72 data points. We used three feature vectors for each image. The first one consists of 1024 features and corresponds to the intensity feature, the second contains 3304 features called "LBP features", and the last one, called "Gabor features", consists of 6750 features.
- **ORL [86]**: The ORL database of facial images consists of 400 images. There are 40 people with 10 different images for each person. Four different feature vectors are used to create

this dataset. The first feature vector corresponds to the GIST feature, which consists of 512 features; the second one contains 59 features called "LBP features", the third one consists of 864 features called "HOG features", and the last one called "Centrist features" consists of 254 features. These images were taken at different times, varying the lighting, facial expression (open/closed eyes, smiling/not smiling), and facial details (glasses/no glasses). The background used for all images is dark.

- **Outdoor Scene [87]:** The Outdoor Scene databases consist of 2688 images. These images are divided into 8 groups. We used four feature vectors for each image. The first one consists of 512 features and corresponds to the feature GIST, the second one contains 432 features called "color moment features", the third one consists of 256 features called "HOG features", and the last one called "LBP features" consists of 48 features.
- **MSRCv1 [88]:** The MSRCv1 dataset consists of 210 instances from Microsoft Research in Cambridge. It has five views with seven groups. We used five feature vectors for each image. The first consists of 24 features and corresponds to the color moment feature, the second consists of 512 features called "GIST features", the third consists of 254 features called "CENTRIST features", the fourth consists of 256 features called "local binary pattern", and the last consists of 512 features called "SIFT".
- **NUS [89]:** The NUS dataset is an image dataset consisting of 12 groups of 2400 images extracted from NUS-WIDE. Six feature vectors are used for each image. The first one consists of 64 features and corresponds to the color histogram; the second one contains 144 features called "color moment features", the third one consists of 73 features called "direction histogram features", the fourth one consists of 128 features called "wavelet texture features", the fifth one consists of 255 features called "block-wise color moments features", and the last one called "SIFT" consists of 500 features.
- **BBCSport [90]:** The BBCSport dataset consists of 544 news documents from the BBC Sports website. It contains 2 views with 5 groups. Each document has two feature vectors. The first view consists of 3183 features, and the second view consists of 3203 features.
- **Caltech101 ¹:** Caltech101 is a large image dataset that contains 6 views and 101 different categories. The first view is composed of 48 feature vectors and corresponds to the Gabor feature. The second contains 40 features called "Wavelet moments features", the third

¹<http://www.vision.caltech.edu/ImageDatasets/Caltech101/>

consists of 254 features called "CENTRIST features", the fourth consists of 1984 features called "Histogram of oriented gradients (HOG) features", the fifth consists of 512 features called "GIST features", and the last one called "LBP features" consists of 928 features.

- **Caltech101-7**²: Caltech101-7 contains 6 views and 7 classes extracted from the Caltech101 dataset, which consists of 101 categories. The feature vectors are the same as those used in the large Caltech101 dataset.
- **UCI Digits**³: UCI Digits are datasets with 2000 instances containing 6 views and 9 classes extracted from a collection of Dutch utility maps. The first consists of 240 features and corresponds to Pixel averages in (2×3) windows (Pix), the second contains 76 features called "Fourier coefficients of the character shapes (FOU) features", the third consists of 216 features, called "Profile correlations (FAC) features", the fourth consists of 47 features called "Zernike moment (ZER) features", the fifth consists of 64 features called "KAR features", and the last one, called "MOR features", consists of 6 features.
- **Extended Yale B Face Dataset**⁴: Extended Yale B Face [91] are datasets corresponding to 1774 images of faces for 28 people (groups) with two different views (different poses and lighting conditions). The first view consists of 900 features and corresponds to "LBP features". The second one contains 45 features called "Covariance ch9 gray features".
- **MNIST** [92]: MNIST is an abbreviation for the Modified National Institute of Standards and Technology database, and is a large data set corresponding to handwritten digits. This large dataset contains 60000 images with 10 classes. In our experiments, we extracted 10000 images with 10 classes. We used two feature vectors for each image. The first one consists of 2048 features and corresponds to the "MNIST small Resnet50 Pooling feature", and the second one contains 4096 features called "MNIST small VGG16 FC1".

Besides, two samples of 25000 and 1000 data points is extracted from the MNIST datasets (MNIST-25000) and (MNIST-1000) respectively, and are tested by our algorithm.

- **COVIDx**⁵: This large dataset consists of 13892 samples divided into 5458 instances corresponding to the pneumonia class, 468 instances corresponding to the COVID19 class, and 7966 instances corresponding to the normal class. Three different feature vectors are

²http://www.vision.caltech.edu/Image_Datasets/Caltech101/

³<https://archive.ics.uci.edu/ml/datasets/Multiple+Features>

⁴<http://vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html>

⁵<https://github.com/lindawangg/COVID-Net/blob/master/docs/COVIDx.md>

used for each image. The first and second features correspond to the deep features of ResNet50 and ResNet101 and consist of 2048 features generated from imagenet pre-trained weights. The third feature is the DenseNet169 feature vector, which consists of 1664 features generated from imagenet’s pre-trained weights.

Tables 4.1 and 4.2 provide the particular descriptions of the datasets used in most chapters. Figure 4.1 also illustrates typical images for these datasets, and Figure 4.2 shows examples of lung X-ray images used for the COVIDx dataset with the corresponding labels.

Table 4.1: Description of the datasets.

View	COIL20	ORL	Out-Scene	BBCSport	MSRCv1	NUS
1	Intensity (1024)	GIST (512)	GIST (512)	Intensity (3183)	GIST (512)	SIFT (255)
2	LBP (3304)	LBP (59)	LBP (48)	LBP (3203)	LBP (256)	Edge direction histogram (73)
3	Gabor (6750)	HOG (864)	HOG (256)	-	Color moment (24)	Wavelet texture (128)
4	-	Centrist (254)	Color moment (432)	-	Centrist (254)	Color moment (144)
5	-	-	-	-	SIFT (512)	Color histogram (64)
# Samples	1440	400	2688	544	210	2400
# Classes	20	40	8	5	7	12

Table 4.2: Description of the datasets.

View	Caltech101	UCI Digits	Extended-Yale	MNIST	COVIDx
1	GIST (512)	FOU (76)	Covariance ch9 gray(45)	Resnet50 Pooling (2048)	Resnet50 Pooling (2048)
2	LBP (928)	Pix (240)	LBP (900)	VGG16 FC1 (4096)	Resnet101 Pooling (2048)
3	Gabor (48)	FAC(216)	-	-	Densenet169 Pooling (1664)
4	Centrist (254)	ZER (47)	-	-	-
5	Wavelet moments (40)	KAR (64)	-	-	-
6	HOG (1984)	MOR (6)	-	-	-
# Samples	9144	2000	1774	10000	13892
# Classes	101	9	28	10	3

We also used three synthetic datasets: Tetra, Hepta, and Chainlink. They were selected from the Fundamental Clustering Problem Suite (FCPS). Only one view is considered for these datasets.

Tetra contains 400 3D points divided into four groups. Hepta contains 212 3D points grouped into seven well-defined clusters with different variances. Chainlink is formed by two clusters that are not linearly separable. It consists of 1000 3D points. These datasets are visualized in Figure 4.3. All these synthetic datasets use 3D data points $\mathbf{p}_i \in \mathbb{R}^3$. The 3-dimensional datasets are transformed into high-dimensional datasets $\mathbf{x}_i \in \mathbb{R}^{100}$ using the following linear and nonlinear mappings $\mathbf{x}_i = \sigma(\mathbf{U} \sigma(\mathbf{W} \mathbf{p}_i))$ where the sigmoid function σ is used to introduce nonlinearity, $\mathbf{W} \in \mathbb{R}^{10 \times 3}$ and $\mathbf{U} \in \mathbb{R}^{100 \times 10}$ are two matrices whose entries follow the Gaussian distribution with zero-mean unit variance independent identically distributed (i.i.d.).

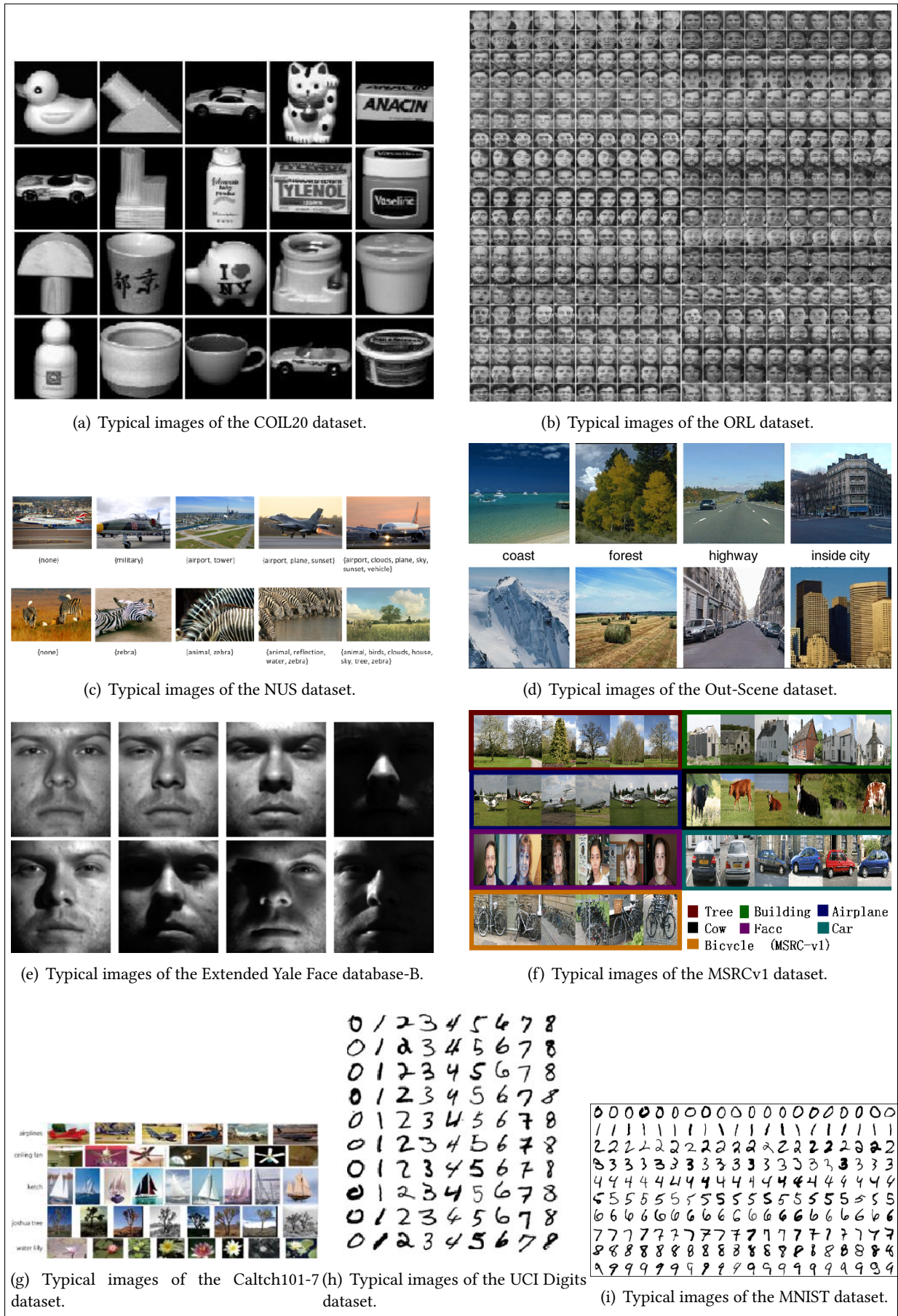


Figure 4.1: Typical images in different datasets.

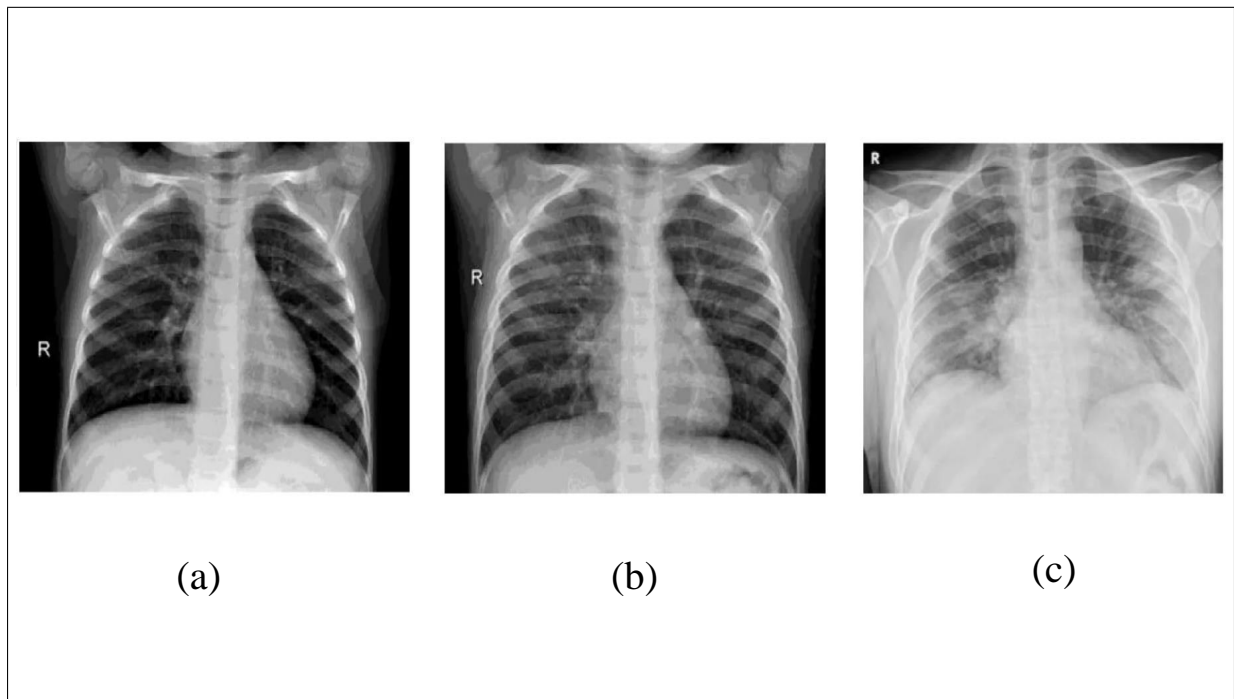


Figure 4.2: Lung X-Rays images for the three mentioned classes: (a): Normal, (b) Pneumonia, and (c) Covid-19.

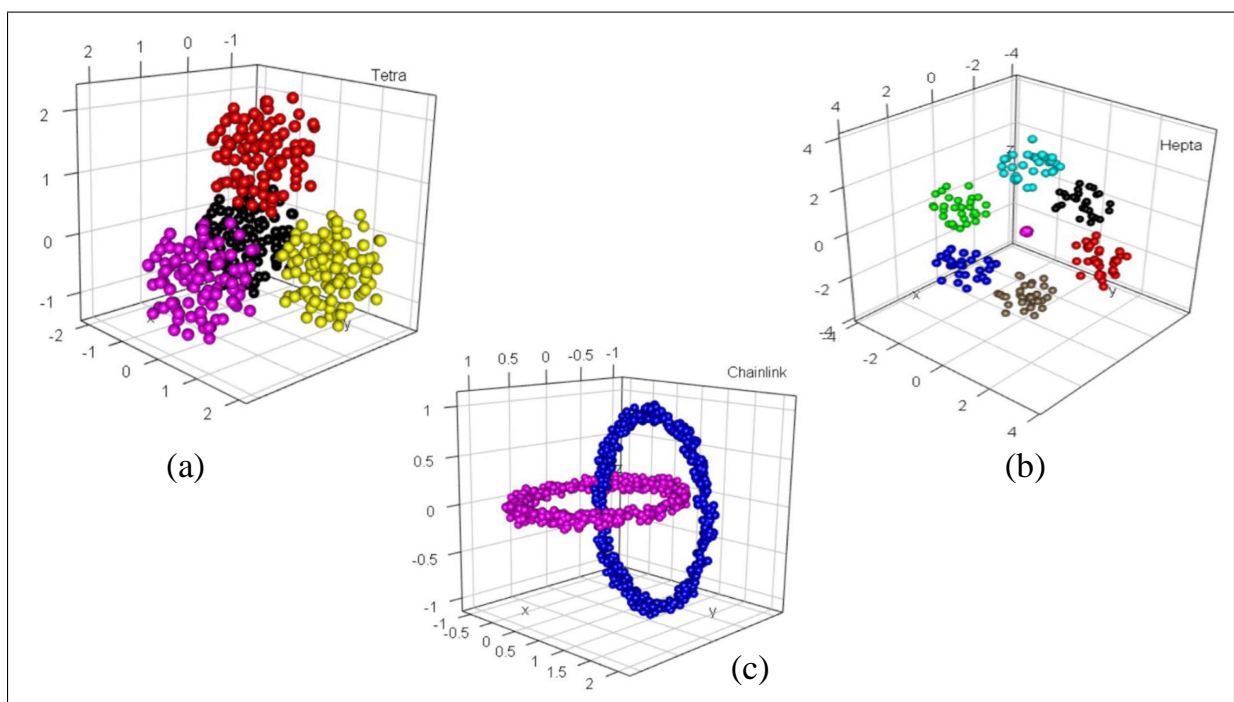


Figure 4.3: Visualization of the original synthetic datasets: (a) Tetra, (b) Hepta, and (c) Chainlink [10].

4.3 Experimental Setup

In our criteria, some methods use the kernel function to map the datasets into a space where they are linearly separable. For this purpose, the Gaussian kernel function was used to construct

the kernel matrix of each view. This nonlinear kernel is widely used in machine learning and has proven to be a good choice for nonlinear data. For a pair of data samples in a given view, the entry K_{ij} is given by $K_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2^{T_0} \sigma_0^2}\right)$. σ_0^2 is set to the average distance between pairs of samples in the considered view and T_0 varies in the set of $\{-4, -2, 0, 2, 4\}$. Note that while our proposed approaches can also be used in cases where each view can have multiple kernel matrices (by changing the type and parameters), our experiments are performed with a single kernel matrix for each view (descriptor). This is to ensure a fair comparison with many methods using the benchmark datasets. The parameter T_0 is used to control the scaling of the Gaussian kernel by a factor 2^{T_0} . Empirically, we found that its optimal value is 2 in most cases, so we fixed it for all datasets.

To initialize the similarity and spectral projection matrices in our methods, we use the same method as in [11, 93], which constructs the similarity matrix of each view according to a smoothing constraint, an ℓ_2 regularization term, and a non-negativity constraint. In this method, \mathbf{x}_i and \mathbf{x}_j have a higher affinity value when the ℓ_2 distance between two data points is small. Moreover, the obtained similarity matrix has exactly m nonzero values, which makes it a sparse matrix, resulting in lower computational cost for graph-based learning. In our experiments, we found that the optimal value is 10 in most cases, so we fixed it for all datasets.

In our experiments, the range for each parameter is chosen to encompass a wide range. This ensures that the optimal values for these parameters are within this range. We used a grid search method to select the values within these ranges. Grid search is a method for exhaustively searching a manually defined subset of the parameter space of a given algorithm. The number of parameters of the algorithm is the spatial dimension of the grid. So, in our case, the grid is in a 3D space. This method starts with the creation of the grid and the selection of the predefined regions. Then, for each parameter combination (represented by the nodes of the grid), a model is created to find the best parameter combination that gives the best clustering performance. The best clustering result is indicated by a cluster performance metric.

For a fair comparison, the best performances for all competing methods are presented by using the authors' source codes with the default or proposed parameter settings (SC, MVCSK, NESE, S-MVSC, CI-GMVC, MCLES), or by directly reproducing the best experimental results from the corresponding published papers (AWP, MLAN, SwMC, AMGL, AASC, MVGL, CorSC, CotSC).

We perform our experiments using Matlab 2018b on a HP PC computer with 4 Intel i-7 8550U processors. The installed memory (RAM) is 12.0 GB (11.9 GB usable). It is also a 64-bit operating system and an x64-based processor.

Direct Multi-view Spectral Clustering with Consistent Kernelized Graph and Convolved Nonnegative Representation

Recently, multi-view clustering has received much attention in the fields of machine learning and pattern recognition. Spectral clustering for single and multiple views has been the common solution. Despite its good clustering performance, it has a major limitation: it requires an extra step of clustering. This extra step, which could be the famous k-means clustering, depends heavily on initialization, which may affect the quality of the clustering result. To address this issue, in this chapter we present a new method called "Multi-view Clustering via Kernelized Graph and Nonnegative Embedding" (MKGNE).

This method can solve four subtasks simultaneously. It provides the unified similarity matrix over all views using kernel tricks; the unified spectral representation matrix, the cluster indicator matrix; and the weight of each view without any other parameters. Moreover, our approach has two interesting properties that the recent algorithms do not have. The first is independence from a particular clustering algorithm. The second feature is the direct estimation of a unified cluster index from the different kernel matrices.

Furthermore, our approach maintains two interesting characteristics that the current methods [11] and [24] do not have simultaneously. The first characteristic is the non-dependence on a specific clustering algorithm. The second characteristic of the proposed method is the joint estimation of

a coherent unified graph, a unified spectral projection, and a unified cluster index matrix. Thus, the suggested method preserves the benefits of both graph-based and matrix factorization-based techniques. The following are the main contributions of this work.

1. We introduce a new multi-view spectral clustering approach that jointly gives: the consistent similarity matrix, the consistent spectral projection matrix, the nonnegative embedding matrix (i.e., the cluster indices of the data), and the weight of each view automatically.
2. Learning of the consistent graph accounts for the underlying correlations from many views by using a kernel representation of the views. As a result, the suggested model effectively searches for nonlinear interactions.
3. The suggested learning approach incorporates learning the cluster indices matrix, which is the result of convolution of the consistent spectral projection matrix across the consistent graph.
4. The presented method avoids any post-process clustering stage, which eliminates the requirement for clustering to be performed many times.

5.1 Proposed Approach

In this chapter, we proposed a novel approach which combines the advantages of graph learning methods and matrix factorization methods. MKGNE achieves the clustering results without any additional step. Figure 5.1 shows an illustration of our proposed multi-view clustering method. As mentioned before, the proposed method can simultaneously estimate 1) the consensus similarity matrix, 2) the consensus data representation matrix, and 3) the nonnegative cluster index matrix. Moreover, the weight of each view is automatically updated without any additional parameters.

Given n samples and V views (feature vectors), the data matrix of each view can be represented as $\mathbf{X}^v = [\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v] \in \mathbb{R}^{d^v \times n}$, where d^v represents the number of features in the corresponding view, where $v = 1, \dots, V$. The corresponding kernel matrices are denoted by \mathbf{K}^v . The dataset is to be grouped into C clusters based on the V views. The unknown matrices are $\mathbf{S} \in \mathbb{R}^{n \times n}$, $\mathbf{P} \in \mathbb{R}^{n \times C}$, and $\mathbf{H} \in \mathbb{R}^{n \times C}$. Our proposed method estimates these matrices simultaneously by integrating several properties such as graph construction using self-representation of data, smoothness of cluster labels, and spectral data convolution. Thus, our proposed criterion has three main terms. To obtain the first term of our proposed criterion, we used the idea of MVCSK method in [24]. To estimate a consistent graph matrix, this method exploits the property of the data to express

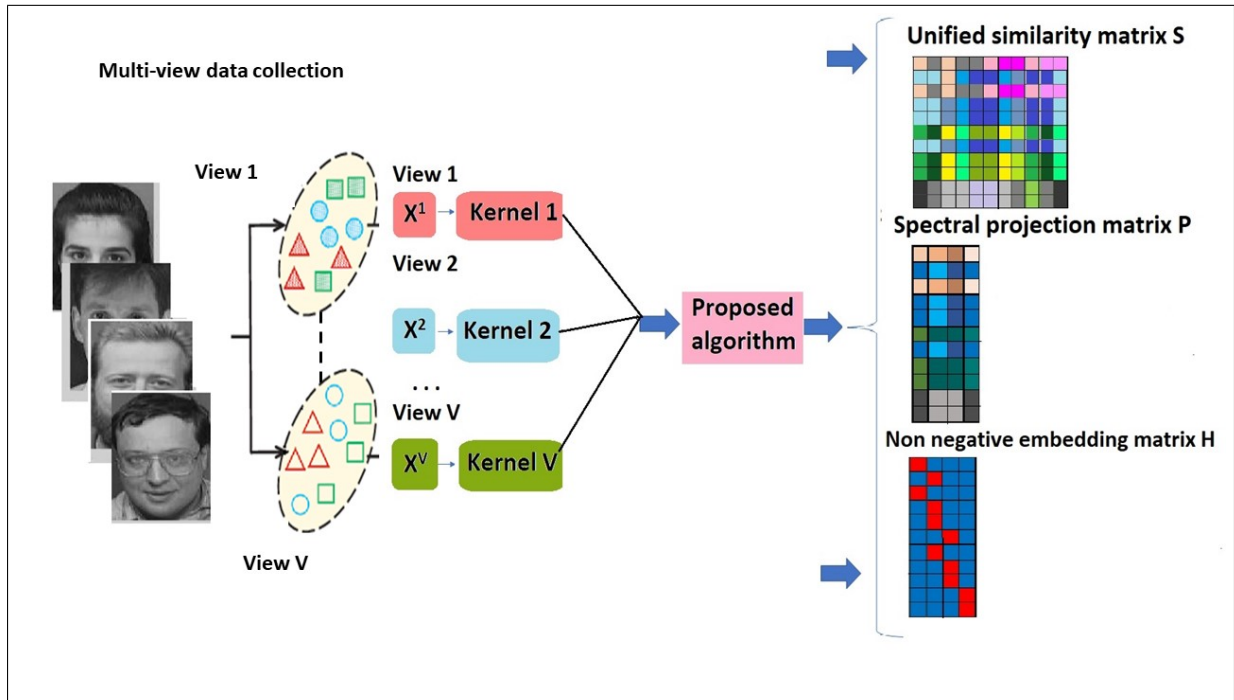


Figure 5.1: Illustration of the MKGNE method.

itself, where the data is mapped nonlinearly. Therefore, the consistent graph matrix \mathbf{S} should satisfy the condition $\min_{\mathbf{S}} \sum_{v=1}^V \|\Phi(\mathbf{X}^v) - \Phi(\mathbf{X}^v) \mathbf{S}\| = \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})}$, where $\mathbf{K}^v = \Phi(\mathbf{X}^v)^T \Phi(\mathbf{X}^v)$ and $\Phi(\cdot)$ is a given nonlinear mapping, which should not be explicitly stated, since only the knowledge of the kernel matrix \mathbf{K}^v is needed. Moreover, to avoid the trivial solution of the consistent graph matrix, a regularization term is used to control the values in this matrix. The first term is as follows:

$$\min_{\mathbf{S}} \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})} + \alpha \|\mathbf{S}\|_2^2 \quad \text{s.t.} \quad \mathbf{S} \geq 0. \quad (5.1)$$

It is also important to assign a weight parameter to each view to represent the contribution of each view to the clustering process. The square root in Eq. (5.1) is used to automatically update the weight of each view [24]. The weight of the view w_v is given by:

$$w_v = \frac{1}{2 \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})}}. \quad (5.2)$$

By using the weight expression in Eq. (5.2), it can be shown that problem (5.1) is equivalent to the following problem:

$$\min_{\mathbf{S}} \sum_{v=1}^V w_v \text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \alpha \|\mathbf{S}\|_2^2 \quad \text{s.t.} \quad \mathbf{S} \geq 0. \quad (5.3)$$

Moreover, inspired by the MVCSK method the first three terms of our method are given below:

$$\min_{\mathbf{S}, \mathbf{P}} \sum_{v=1}^V w_v \text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) \quad s.t. \quad \mathbf{S} \geq 0, \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (5.4)$$

In [24], the authors proposed an optimization algorithm to estimate the two unknown matrices. However, the resulting graph and spectral embedding need an additional step in order to retrieve the clusters, meaning that the clustering should be possibly carried out using several trials.

In order to overcome this limitation, we propose a model that integrates a nonnegative matrix that directly gives the clusters' membership. This matrix should be learned jointly with the unified graph and the corresponding spectral projection. To this end, we will assume that this embedding matrix is the result of blending (convolving) the spectral representation of the instances over the graph. In detail, let \mathbf{h}_i denote the cluster indices (a row vector) associated with the i -th instance. We will assume that the row vector is given by the following approximation:

$$\mathbf{h}_i \approx \mathbf{S}_{i,*} \mathbf{P} \quad s.t. \quad \mathbf{h}_i \geq 0, \quad (5.5)$$

where $\mathbf{S}_{i,*}$ denotes the i -th row of the graph matrix \mathbf{S} . Since the cluster index of the i -th instance corresponds to the largest element in the row vector \mathbf{h}_i , we can conclude that the cluster index corresponds to the maximum similarity (scalar product) between the edges linked to that instance and each column vector in the spectral projection matrix \mathbf{P} . Also, Eq. (5.5) stipulates that the cluster embedding is nothing but the convolution of the spectral features over the consistent graph [94].

In a matrix form, Eq. (5.5) can be written as:

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{S} \mathbf{P}\|_F^2 \quad s.t. \quad \mathbf{H} \geq 0. \quad (5.6)$$

Since the matrix \mathbf{P} is orthogonal, the above equation can be written as:

$$\min_{\mathbf{H}} \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_F^2 \quad s.t. \quad \mathbf{H} \geq 0. \quad (5.7)$$

Finally, our proposed learning model becomes:

$$\begin{aligned} \min_{\mathbf{S}, \mathbf{P}, \mathbf{H}} \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})} + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \\ \text{s.t. } \mathbf{S} \geq 0, \mathbf{P}^T \mathbf{P} = \mathbf{I}, \mathbf{H} \geq 0, \end{aligned} \quad (5.8)$$

where μ , λ_1 , and λ_2 are positive regularization parameters. By adopting the weight given by equation (5.2), problem (5.8) is equivalent to the following problem:

$$\begin{aligned} \min_{\mathbf{S}, \mathbf{P}, \mathbf{H}} \sum_{v=1}^V w_v \text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \\ \text{s.t. } \mathbf{S} \geq 0, \mathbf{P}^T \mathbf{P} = \mathbf{I}, \mathbf{H} \geq 0. \end{aligned} \quad (5.9)$$

In the proposed model, the unified graph matrix \mathbf{S} , is linked to the view-based kernel matrices (through the first term), and at the same time, is linked to the unified spectral representation \mathbf{P} , and the cluster index matrix \mathbf{H} through the third and fourth terms.

Once the matrix \mathbf{H} is estimated, we assign the i -th sample to the cluster corresponding to the index of the highest element in the i -th row of \mathbf{H} .

5.1.1 Optimization

In this section, we will detail the procedure used to optimize the objective function of MKGNE over the three unknown matrices: \mathbf{S} , \mathbf{H} , and \mathbf{P} . This optimization is solved iteratively. We adopt an alternating optimization scheme. In other words, we fix two matrices and update the remaining matrix. This process is repeated until convergence.

Update \mathbf{H} :

To get the matrix \mathbf{H} we fix the variables \mathbf{P} , \mathbf{S} and w_v ($v = 1, \dots, V$), By removing the irrelevant terms, the resulting problem becomes:

$$\min_{\mathbf{H}} \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \quad \text{s.t. } \mathbf{H} \geq 0 \quad (5.10)$$

Since $\mathbf{P}^T \mathbf{P} = \mathbf{I}$, the above problem will be equivalent to:

$$\min_{\mathbf{H}} \|\mathbf{S} \mathbf{P} - \mathbf{H}\|_2^2 \quad \text{s.t. } \mathbf{H} \geq 0 \quad (5.11)$$

The optimal solution to (5.11) is given by:

$$\mathbf{H} = \max(\mathbf{S} \mathbf{P}, \mathbf{0}). \quad (5.12)$$

Thus, the matrix \mathbf{H} is the element-wise ReLU (Rectified Linear Unit) operator applied on the elements of the matrix $\mathbf{S}\mathbf{P}$.

Update \mathbf{P} :

To get the matrix \mathbf{P} we fix the variables \mathbf{H} , \mathbf{S} and w_v . Thus, problem (5.8) will be equivalent to:

$$\min_{\mathbf{P}} Tr(\mathbf{P}^T \mathbf{L} \mathbf{P}) + \frac{\lambda_2}{\lambda_1} \|\mathbf{S} - \mathbf{H}\mathbf{P}^T\|_2^2 \quad s.t. \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (5.13)$$

To solve Equation (5.13), the following equation is used:

$$\|\mathbf{S}\mathbf{P} - \mathbf{H}\|_2^2 = Tr[(\mathbf{S}\mathbf{P} - \mathbf{H})^T (\mathbf{S}\mathbf{P} - \mathbf{H})]. \quad (5.14)$$

The derivative of the functional in (5.13) w.r.t. \mathbf{P} is given by: $\frac{\partial f}{\partial \mathbf{P}} = 2\mathbf{L}\mathbf{P} + 2\frac{\lambda_2}{\lambda_1}\mathbf{S}^T\mathbf{S}\mathbf{P} - 2\frac{\lambda_2}{\lambda_1}\mathbf{S}^T\mathbf{H}$.

The optimal \mathbf{P} can be obtained by vanishing this derivative. Thus, \mathbf{P} will be given by:

$$\mathbf{P} = \left(\mathbf{L} + \frac{\lambda_2}{\lambda_1} \mathbf{S}^T \mathbf{S} \right)^{-1} \frac{\lambda_2}{\lambda_1} \mathbf{S}^T \mathbf{H}. \quad (5.15)$$

In order to satisfy the orthogonality constraint, an orthogonalization step is applied on the obtained \mathbf{P} .

Update \mathbf{S} :

To get the matrix \mathbf{S} we fix the variables \mathbf{P} , \mathbf{H} and w_v . We minimize the resulting functional over the matrix \mathbf{S} . This functional is given by:

$$g = \sum_{v=1}^V w_v Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \mu \|\mathbf{S}\|_2^2 + \lambda_1 Tr(\mathbf{P}^T \mathbf{L} \mathbf{P}) + \lambda_2 \|\mathbf{S} - \mathbf{H}\mathbf{P}^T\|_2^2 \quad s.t. \quad \mathbf{S} \geq 0. \quad (5.16)$$

In the spectral clustering analysis, we have the following well-known identity:

$$Tr(\mathbf{P}^T \mathbf{L} \mathbf{P}) = \frac{1}{2} \sum_i \sum_j \|\mathbf{p}_i - \mathbf{p}_j\|^2 S_{ij}, \quad (5.17)$$

where \mathbf{p}_i denotes the i -th row of the matrix \mathbf{P} . Let $\mathbf{Q} \in \mathbb{R}^{n \times n}$ denote the symmetric matrix of pair distances associated the rows of \mathbf{P} , i.e. $Q_{ij} = \frac{1}{2} \|\mathbf{p}_i - \mathbf{p}_j\|^2$. By using the definition of the matrix \mathbf{Q} , Eq. (5.17) becomes:

$$Tr(\mathbf{P}^T \mathbf{L} \mathbf{P}) = \frac{1}{2} \sum_i \sum_j \|\mathbf{p}_i - \mathbf{p}_j\|^2 S_{ij} = Tr(\mathbf{Q}\mathbf{S}). \quad (5.18)$$

By plugging Eq. (5.18) into Equation (5.16), the latter becomes:

$$g = \sum_{v=1}^V w_v \text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{Q} \mathbf{S}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \quad s.t. \quad \mathbf{S} \geq 0. \quad (5.19)$$

The graph matrix \mathbf{S} can be obtained by vanishing the derivative of g (Eq. (5.19)) w.r.t. \mathbf{S} . This yields the expression of \mathbf{S} :

$$\mathbf{S} = \text{ReLU} \left\{ \left(2 \sum_{v=1}^V w_v \mathbf{K}^v + 2(\mu + \lambda_2) \mathbf{I} \right)^{-1} \left(2 \sum_{v=1}^V w_v \mathbf{K}^v + 2\lambda_2 \mathbf{H} \mathbf{P}^T - \lambda_1 \mathbf{Q} \right) \right\}. \quad (5.20)$$

Update w_v : The weights w_v are updated using Eq. (5.2).

The main steps of our proposed method are illustrated in **Algorithm 1**.

Algorithm 1 MKGNE

Input: Data samples in V views $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$.
The spectral embedding matrices \mathbf{P}^v , $v = 1, \dots, V$.
The similarity matrices \mathbf{S}^v , $v = 1, \dots, V$.
Parameters μ , λ_1 , λ_2 .

Output: The graph matrix \mathbf{S} .
The spectral projection matrix \mathbf{P} .
The consistent non negative cluster index matrix \mathbf{H} .

Initialization:

The weight of each view $w_v = \frac{1}{V}$.
Compute the kernel matrix \mathbf{K}^v for each view.
Initialize \mathbf{S} (\mathbf{P}) by taking the average of the matrices \mathbf{S}^v (\mathbf{P}^v).

Repeat

Step 1. Update \mathbf{H} using Eq. (5.12).

Step 2. Update \mathbf{P} using Eq. (5.15).

Step 3. Update \mathbf{S} using Eq. (5.20).

Step 4. Update w_v using Eq. (5.2).

End

From the depicted steps of **Algorithm 1**, it can be seen that initial values for the matrices \mathbf{S} and \mathbf{P} are needed. To initialize these two matrices, the efficient graph construction method [93] is invoked on the data of each view. Therefore, the average of the different graphs obtained for the different views will be used as the initial \mathbf{S} . Similarly, the average of the different spectral

embeddings will be used as the initial \mathbf{P} . This initialization scheme is similar to the one adopted in [11]. However, the latter method does not update each individual graphs nor generate a unified graph.

5.1.2 Differences with state-of-the-art methods

The most related state-of-the-art methods are MVSC [24] and NESE [11]. The former estimates a unified graph and a unified spectral projection. However, it does not estimate a clustering matrix, and needs an extra clustering step. The latter only provides a unified clustering matrix from view-based graphs and spectral projections. On the other hand, our proposed approach jointly estimates a unified graph, a unified spectral representation and the corresponding cluster index matrix.

5.1.3 Convergence analysis

In this section, we will prove that the objective function of **Algorithm 1** is a non-increasing over the iterations. This ensures its convergence.

Let's begin by introducing an important Lemma.

Lemma 1. For any two positive constants c and t , we have the following inequality:

$$\sqrt{c} - \frac{c}{2\sqrt{t}} \leq \sqrt{t} - \frac{t}{2\sqrt{t}}. \quad (5.21)$$

Proof of Lemma 1:

$$(\sqrt{c} - \sqrt{t})^2 \geq 0 \implies (\sqrt{c})^2 - 2\sqrt{c}\sqrt{t} + (\sqrt{t})^2 \geq 0 \implies 2\sqrt{c}\sqrt{t} - (\sqrt{c})^2 \leq 2(\sqrt{t})^2 - (\sqrt{t})^2 \implies (5.21).$$

Let the scalar function $f(\mathbf{S})$ denotes the following entity:

$$f(\mathbf{S}) = \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T (\mathbf{D} - \mathbf{S}) \mathbf{P}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2.$$

Let $\mathbf{F}^v(\mathbf{S}) = \mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}$.

Using the above definitions for $f(\mathbf{S})$ and $\mathbf{F}^v(\mathbf{S})$ in problem (5.9), the latter can be written in a more compact form:

$$\min \sum_{v=1}^V w_v \text{Tr}(\mathbf{F}^v(\mathbf{S})) + f(\mathbf{S}). \quad (5.22)$$

Suppose that $\hat{\mathbf{S}}$ is the solution of step 3 in **Algorithm 1**, and \mathbf{S} is the solution obtained at the previous iteration. We like to prove the following inequality:

$$\sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v(\hat{\mathbf{S}}))} + f(\hat{\mathbf{S}}) \leq \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v(\mathbf{S}))} + f(\mathbf{S}). \quad (5.23)$$

Obviously, if (5.23) is satisfied for any two consecutive iterations of Algorithm 1, the objective function of the original problem (5.8) is non-increasing over the iterations since each one consists of three minimization steps.

Proof of (5.23):

According to the definition of $\hat{\mathbf{S}}$, we have:

$$\sum_{v=1}^V w_v \text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}})) + f(\hat{\mathbf{S}}) \leq \sum_{v=1}^V w_v \text{Tr}(\mathbf{F}^v(\mathbf{S})) + f(\mathbf{S}) \quad (5.24)$$

By plugging the expression of w_v (i.e., Eq. (5.2)), into Eq. (5.24) this one becomes:

$$\sum_{v=1}^V \frac{\text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}}))}{2\sqrt{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} + f(\hat{\mathbf{S}}) \leq \sum_{v=1}^V \frac{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}{2\sqrt{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} + f(\mathbf{S}). \quad (5.25)$$

By using **Lemma 1** and setting $c = \text{Tr}(\mathbf{F}^v(\mathbf{S}))$ and $t = \text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}}))$, we get the following:

$$\sum_{v=1}^V \left\{ \sqrt{\frac{\text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}}))}{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} - \frac{\text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}}))}{2\sqrt{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} \right\} \leq \sum_{v=1}^V \left\{ \sqrt{\frac{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} - \frac{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}{2\sqrt{\text{Tr}(\mathbf{F}^v(\mathbf{S}))}} \right\}. \quad (5.26)$$

By adding the left and right sides in Equations (5.25) and (5.26), we will obtain:

$$\sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{F}^v(\hat{\mathbf{S}}))} + f(\hat{\mathbf{S}}) \leq \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{F}^v(\mathbf{S}))} + f(\mathbf{S}), \quad (5.27)$$

which proves the convergence of **Algorithm 1**.

5.2 Performance Evaluation

5.2.1 Datasets

In order to verify the effectiveness of the proposed approach, we use nine datasets with different data types and sizes: COIL20, ORL, Outdoor Scene, NUS, MSRCv1, BBCSport, Caltech101-7, Extended Yale B Face, and MNIST.

5.2.2 Experimental Setup

Clustering methods may use different paradigms as well as different objective functions, leading to different properties. We compare the performance of the proposed multi-view clustering approach with that of several multi-view clustering methods:

- Co-training approach for multi-view spectral clustering "CotSC" [35].
- Co-regularized approach for multi-view spectral clustering "CorSC" [34].
- Multi-view clustering and semi-supervised classification with adaptive neighbours "MLAN" [44].
- Self-weighted multi-view clustering with multiple graphs "SwMC" [95].
- Affinity aggregation for spectral clustering "AASC" [96].
- Graph learning for multi-view clustering "MVGL" [63].
- Parameter-free auto-weighted multiple graph learning "AMGL" [13].
- Multi-view clustering via adaptively weighted Procrustes "AWP" [20].
- Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding "NESE" [11].
- Auto-weighted multi-view clustering via kernelized graph learning "MVCSK" [24].
- Multi-view spectral clustering via sparse graph learning "S-MVSC" [77].
- Consistency-aware and Inconsistency-aware Graph-based Multi-View Clustering "CI-GMVC" [78]
- Multi-view spectral clustering via constrained nonnegative embedding "CNESE" [97].

We also include the classic spectral clustering on a single view. We report the results obtained by the best single view (SC-best). We present two groups of comparison. In the first group, we provide a comparison based on the above list of clustering methods on the COIL20, ORL, Outdoor Scene and NUS datasets. In the second group of experiments, we compare our proposed method with the most related state-of-the-art methods, namely the MVSC, NESE, S-MVSC, CI-GMVC, and CNESE methods.

A data normalization step is performed before any computation. For each view, a gaussian kernel matrix is generated.

The initialization of our method is carried out using the same initialization method employed in [11]. Once all individual graphs and spectral projections are computed for each view, we compute an average of them. These average matrices will be used as an initial guess for the matrices \mathbf{S} and \mathbf{P} . In our approach, three parameters are used: λ_1 , λ_2 , and μ . λ_1 and λ_2 are searched in the set $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, while μ is chosen from $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$.

5.2.3 Experimental results

In this section, we present the experimental results obtained with our MKGNE method. We compare the obtained results with different clustering methods. Table 5.1 shows the performance indicators for fourteen competing clustering methods. The results were obtained using the COIL20, ORL, Out-Scene, and NUS datasets. The values in bold correspond to the best performance obtained for each dataset. The experimental results presented in Table 5.1 show that the best clustering methods include MVSC, NESE, S-MVSC, CI-GMVC, CNESE and MKGNE (proposed). These methods will mainly be used for method comparison in the remaining experiments.

Table 5.2 presents a comparison between our method and the five competing methods MVSC, NESE, S-MVSC, CI-GMVC and CNESE. The datasets used are: BBCSport, MSRCv1, Caltech7, Extended Yale B face, MNIST and MNIST-1000 datasets.

5.2.4 Parameter sensitivity

In this section, we study the sensitivity of the parameters and their impact on the clustering performance of the proposed method. The approach has four parameters: λ_1 , λ_2 , μ , and T_0 . The latter is used to control the scaling of the Gaussian kernel by a factor of 2^{T_0} . Empirically, we found that its optimal value is 2 in most cases. Therefore, we set it to 2 in the current study. The values of λ_1 and λ_2 are chosen from the set $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, while μ is chosen from $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$.

Figures 5.2 and 5.3 show the clustering performance as a function of these parameters for the datasets ORL and MSRCv1, respectively. The subfigures 5.2(a) and 5.2(b), corresponding to the ORL dataset, show the indicators ACC and NMI as a function of the parameters λ_1 and μ , and with the fixed parameter λ_2 . According to these two subfigures, the best results for ACC and NMI are 90% and 97%, respectively. These correspond to values of $\mu = 0.05$, $\lambda_1 = 10^{-3}$ and $\lambda_2 = 10^{-5}$. The subfigures 5.2(c) and 5.2(d) show the indicators ACC and NMI as a function of the parameter λ_2 , while the other two parameters λ_1 and μ are fixed. According to these two subfigures, it is preferable that λ_2 is between 10^{-6} and 10^{-4} to obtain better results.

According to the two subfigures 5.3(a) and 5.3(b) corresponding to the MSRCv1 dataset, the best results for ACC and NMI are 83% and 79%, respectively. These correspond to values of $\mu = 0.3$, $\lambda_1 = 10^{-3}$ and $\lambda_2 = 10^{-4}$. The subfigures 5.3(c) and 5.3(d) show the ACC and NMI as a function of the parameter λ_2 , while the other two parameters λ_1 and μ are fixed. According to these subfigures, it is preferable that λ_2 is between 10^{-4} and 10^{-5} to obtain better results.

Table 5.1: Clustering performance on the COIL20, ORL, Out-Scene, and NUS datasets.

Dataset	Method	ACC	NMI	Purity	ARI
COIL20	SC-Best [12]	0.73 (± 0.01)	0.82 (± 0.01)	0.75 (± 0.01)	0.68 (± 0.02)
	AWP [20]	0.68 (± 0.00)	0.87 (± 0.00)	0.75 (± 0.00)	0.71 (± 0.00)
	MLAN [44]	0.84 (± 0.00)	0.92 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)
	SwMC [95]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.84 (± 0.00)
	AMGL [13]	0.80 (± 0.04)	0.91 (± 0.02)	0.85 (± 0.03)	0.74 (± 0.07)
	AASC [96]	0.79 (± 0.00)	0.89 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MVGL [63]	0.78 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)	0.75 (± 0.00)
	CorSC [34]	0.68 (± 0.04)	0.78 (± 0.02)	0.70 (± 0.03)	0.62 (± 0.03)
	CotSC [35]	0.70 (± 0.03)	0.80 (± 0.02)	0.72 (± 0.03)	0.65 (± 0.03)
	NESE [11]	0.77 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.69 (± 0.00)
	MVCSK [24]	0.65 (± 0.04)	0.80 (± 0.02)	0.70 (± 0.03)	0.61 (± 0.05)
	S-MVSC [77]	0.62 (± 0.01)	0.86 (± 0.02)	0.77 (± 0.02)	0.97 (± 0.02)
	CI-GMVC [78]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)
	CNESE [97]	0.82 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.78 (± 0.00)
MKGNE	0.95 (± 0.00)	0.99 (± 0.00)	0.95 (± 0.00)	0.95 (± 0.00)	
ORL	SC-Best [12]	0.66 (± 0.02)	0.76 (± 0.02)	0.71 (± 0.02)	0.67 (± 0.01)
	AWP [20]	0.80 (± 0.00)	0.91 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MLAN [44]	0.78 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.67 (± 0.00)
	SwMC [95]	0.77 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)	0.62 (± 0.00)
	AMGL [13]	0.75 (± 0.02)	0.90 (± 0.02)	0.82 (± 0.02)	0.63 (± 0.09)
	AASC [96]	0.82 (± 0.02)	0.91 (± 0.01)	0.85 (± 0.01)	0.76 (± 0.02)
	MVGL [63]	0.75 (± 0.00)	0.88 (± 0.00)	0.80 (± 0.00)	0.55 (± 0.00)
	CorSC [34]	0.77 (± 0.03)	0.90 (± 0.01)	0.82 (± 0.03)	0.72 (± 0.04)
	CotSC [35]	0.75 (± 0.04)	0.87 (± 0.01)	0.78 (± 0.03)	0.67 (± 0.03)
	NESE [11]	0.82 (± 0.00)	0.91 (± 0.00)	0.85 (± 0.00)	0.75 (± 0.00)
	MVCSK [24]	0.85 (± 0.02)	0.94 (± 0.01)	0.88 (± 0.02)	0.81 (± 0.02)
	S-MVSC [77]	0.80 (± 0.02)	0.93 (± 0.01)	0.82 (± 0.02)	0.89 (± 0.01)
	CI-GMVC [78]	0.81 (± 0.00)	0.92 (± 0.00)	0.85 (± 0.00)	0.74 (± 0.00)
	CNESE [97]	0.87 (± 0.00)	0.95 (± 0.00)	0.89 (± 0.00)	0.84 (± 0.00)
MKGNE	0.91 (± 0.00)	0.97 (± 0.00)	0.93 (± 0.00)	0.89 (± 0.00)	
Out-Scene	SC-best [12]	0.47 (± 0.01)	0.39 (± 0.01)	0.57 (± 0.01)	0.34 (± 0.01)
	AWP [20]	0.65 (± 0.00)	0.51 (± 0.00)	0.65 (± 0.00)	0.42 (± 0.00)
	MLAN [44]	0.55 (± 0.02)	0.47 (± 0.01)	0.55 (± 0.02)	0.33 (± 0.03)
	SwMC [95]	0.50 (± 0.00)	0.47 (± 0.00)	0.50 (± 0.00)	0.38 (± 0.00)
	AMGL [13]	0.51 (± 0.05)	0.45 (± 0.03)	0.52 (± 0.04)	0.34 (± 0.05)
	AASC [96]	0.60 (± 0.00)	0.48 (± 0.00)	0.60 (± 0.00)	0.35 (± 0.00)
	MVGL [63]	0.42 (± 0.00)	0.31 (± 0.00)	0.43 (± 0.00)	0.16 (± 0.00)
	CorSC [34]	0.51 (± 0.04)	0.39 (± 0.03)	0.52 (± 0.03)	0.31 (± 0.02)
	CotSC [35]	0.38 (± 0.02)	0.22 (± 0.01)	0.39 (± 0.02)	0.16 (± 0.01)
	NESE [11]	0.63 (± 0.00)	0.53 (± 0.00)	0.66 (± 0.00)	0.46 (± 0.00)
	MVCSK [24]	0.65 (± 0.01)	0.52 (± 0.00)	0.65 (± 0.01)	0.42 (± 0.00)
	S-MVSC [77]	0.48 (± 0.01)	0.54 (± 0.02)	0.65 (± 0.01)	0.46 (± 0.04)
	CI-GMVC [78]	0.35 (± 0.01)	0.31 (± 0.00)	0.35 (± 0.01)	0.19 (± 0.00)
	CNESE [97]	0.66 (± 0.00)	0.55 (± 0.00)	0.67 (± 0.00)	0.47 (± 0.00)
MKGNE	0.72 (± 0.00)	0.57 (± 0.00)	0.72 (± 0.00)	0.48 (± 0.00)	
NUS	SC-Best [12]	0.21(± 0.01)	0.09(± 0.01)	0.21(± 0.01)	0.07(± 0.02)
	AWP [20]	0.28(± 0.00)	0.15(± 0.00)	0.29(± 0.00)	0.09(± 0.00)
	MLAN [?]	0.25(± 0.00)	0.15(± 0.00)	0.26(± 0.00)	0.04(± 0.00)
	SwMC [95]	0.15(± 0.00)	0.08(± 0.00)	0.17(± 0.00)	0.01(± 0.00)
	AMGL [13]	0.25(± 0.01)	0.13(± 0.01)	0.27(± 0.01)	0.07(± 0.01)
	AASC [96]	0.25(± 0.00)	0.13(± 0.00)	0.27(± 0.00)	0.06(± 0.00)
	MVGL [63]	0.15(± 0.00)	0.07(± 0.00)	0.16(± 0.00)	0.01(± 0.00)
	CorSC [34]	0.27(± 0.01)	0.14(± 0.01)	0.29(± 0.01)	0.09(± 0.01)
	CotSC [35]	0.29(± 0.01)	0.16(± 0.01)	0.30(± 0.01)	0.09(± 0.01)
	NESE [11]	0.30(± 0.00)	0.17(± 0.00)	0.32(± 0.00)	0.10(± 0.00)
	MVCSK [24]	0.26(± 0.01)	0.15(± 0.00)	0.28(± 0.00)	0.08(± 0.00)
	S-MVSC [77]	0.27 (± 0.02)	0.13 (± 0.01)	0.26 (± 0.02)	0.08 (± 0.02)
	CI-GMVC [78]	0.26 (± 0.01)	0.14 (± 0.01)	0.25 (± 0.01)	0.09 (± 0.00)
	CNESE [97]	0.28 (± 0.00)	0.16 (± 0.00)	0.31 (± 0.00)	0.09 (± 0.00)
MKGNE	0.30 (± 0.00)	0.15 (± 0.00)	0.31 (± 0.00)	0.10 (± 0.00)	

Table 5.2: Clustering performance on the BBCSport, MSRCv1, Caltech101-7, Extended-Yale, MNIST and MNIST-1000 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
BBCSport	MVCSK	0.90 (± 0.07)	0.82 (± 0.02)	0.90 (± 0.02)	0.85 (± 0.07)
	NESE	0.72 (± 0.00)	0.69 (± 0.00)	0.75 (± 0.00)	0.60 (± 0.00)
	S-MVSC	0.58 (± 0.07)	0.67 (± 0.01)	0.73 (± 0.02)	0.83 (± 0.04)
	CI-GMVC	0.61 (± 0.00)	0.46 (± 0.00)	0.63 (± 0.00)	0.36 (± 0.00)
	CNESE	0.72 (± 0.00)	0.68 (± 0.00)	0.76 (± 0.00)	0.60 (± 0.00)
	MKGNE	0.98 (± 0.00)	0.94 (± 0.00)	0.98 (± 0.00)	0.95 (± 0.00)
MSRCv1	MVCSK	0.70 (± 0.02)	0.59 (± 0.03)	0.70 (± 0.02)	0.50 (± 0.04)
	NESE	0.77 (± 0.00)	0.72 (± 0.00)	0.80 (± 0.00)	0.64 (± 0.00)
	S-MVSC	0.60 (± 0.00)	0.69 (± 0.02)	0.74 (± 0.02)	0.79 (± 0.01)
	CI-GMVC	0.74 (± 0.00)	0.72 (± 0.00)	0.77 (± 0.00)	0.59 (± 0.00)
	CNESE	0.86 (± 0.00)	0.76 (± 0.00)	0.86 (± 0.00)	0.72 (± 0.00)
	MKGNE	0.83 (± 0.00)	0.79 (± 0.00)	0.83 (± 0.00)	0.72 (± 0.00)
Caltech101-7	MVCSK	0.57 (± 0.02)	0.51 (± 0.02)	0.83 (± 0.01)	0.45 (± 0.03)
	NESE	0.67 (± 0.00)	0.55 (± 0.00)	0.87 (± 0.00)	0.52 (± 0.00)
	S-MVSC	0.64 (± 0.03)	0.55 (± 0.02)	0.72 (± 0.01)	0.51 (± 0.03)
	CI-GMVC	0.74 (± 0.00)	0.54 (± 0.00)	0.85 (± 0.00)	0.48 (± 0.00)
	CNESE	0.69 (± 0.00)	0.58 (± 0.00)	0.88 (± 0.00)	0.56 (± 0.00)
	MKGNE	0.69 (± 0.00)	0.64 (± 0.00)	0.89 (± 0.00)	0.57 (± 0.00)
Extended-Yale	MVCSK	0.33 (± 0.00)	0.42 (± 0.00)	0.34 (± 0.00)	0.18 (± 0.00)
	NESE	0.43 (± 0.00)	0.58 (± 0.00)	0.47 (± 0.00)	0.25 (± 0.00)
	S-MVSC	0.48 (± 0.03)	0.61 (± 0.01)	0.60 (± 0.01)	0.36 (± 0.05)
	CI-GMVC	0.32 (± 0.00)	0.34 (± 0.00)	0.35 (± 0.00)	0.02 (± 0.00)
	CNESE	0.60 (± 0.00)	0.75 (± 0.00)	0.60 (± 0.00)	0.51 (± 0.00)
	MKGNE	0.81 (± 0.00)	0.85 (± 0.00)	0.82 (± 0.00)	0.72 (± 0.00)
MNIST	MVCSK	0.49 (± 0.00)	0.41 (± 0.00)	0.50 (± 0.00)	0.29 (± 0.00)
	NESE	0.81 (± 0.00)	0.83 (± 0.00)	0.85 (± 0.00)	0.76 (± 0.00)
	S-MVSC	0.77 (± 0.01)	0.81 (± 0.01)	0.81 (± 0.02)	0.76 (± 0.07)
	CI-GMVC	0.66 (± 0.00)	0.71 (± 0.00)	0.71 (± 0.00)	0.51 (± 0.00)
	CNESE	0.81 (± 0.00)	0.83 (± 0.00)	0.86 (± 0.00)	0.78 (± 0.00)
	MKGNE	0.81 (± 0.00)	0.84 (± 0.00)	0.85 (± 0.00)	0.76 (± 0.00)
MNIST-1000	MVCSK	0.70 (± 0.00)	0.61 (± 0.00)	0.70 (± 0.00)	0.52 (± 0.00)
	NESE	0.78 (± 0.00)	0.79 (± 0.00)	0.83 (± 0.00)	0.71 (± 0.00)
	S-MVSC	0.66 (± 0.02)	0.76 (± 0.01)	0.76 (± 0.00)	0.77 (± 0.05)
	CI-GMVC	0.65 (± 0.00)	0.71 (± 0.00)	0.73 (± 0.00)	0.50 (± 0.00)
	CNESE	0.77 (± 0.00)	0.77 (± 0.00)	0.81 (± 0.00)	0.68 (± 0.00)
	MKGNE	0.84 (± 0.00)	0.80 (± 0.00)	0.84 (± 0.00)	0.74 (± 0.00)

5.2.5 Analysis of the results and method comparison

From the above tables and figures, it can be seen that the performances obtained by all multi-view clustering methods are superior to those obtained by the first algorithm (SC -Best), which refers to the best single view that performs spectral clustering. This result is normal since the use of multiple views allows a better use of the data and provides additional information about the data to improve the clustering results.

Moreover, it is clear from Tables 5.1 and 5.2 that the performance of our method is superior or similar to that of other methods for most datasets.

Table 5.1 shows that our proposed approach outperforms other competing methods for the COIL20, ORL, and Out-Scene datasets (except for the ARI indicator in the COIL20 dataset). For the NUS dataset, the result of our method is slightly lower than that of NESE method, but it is higher than

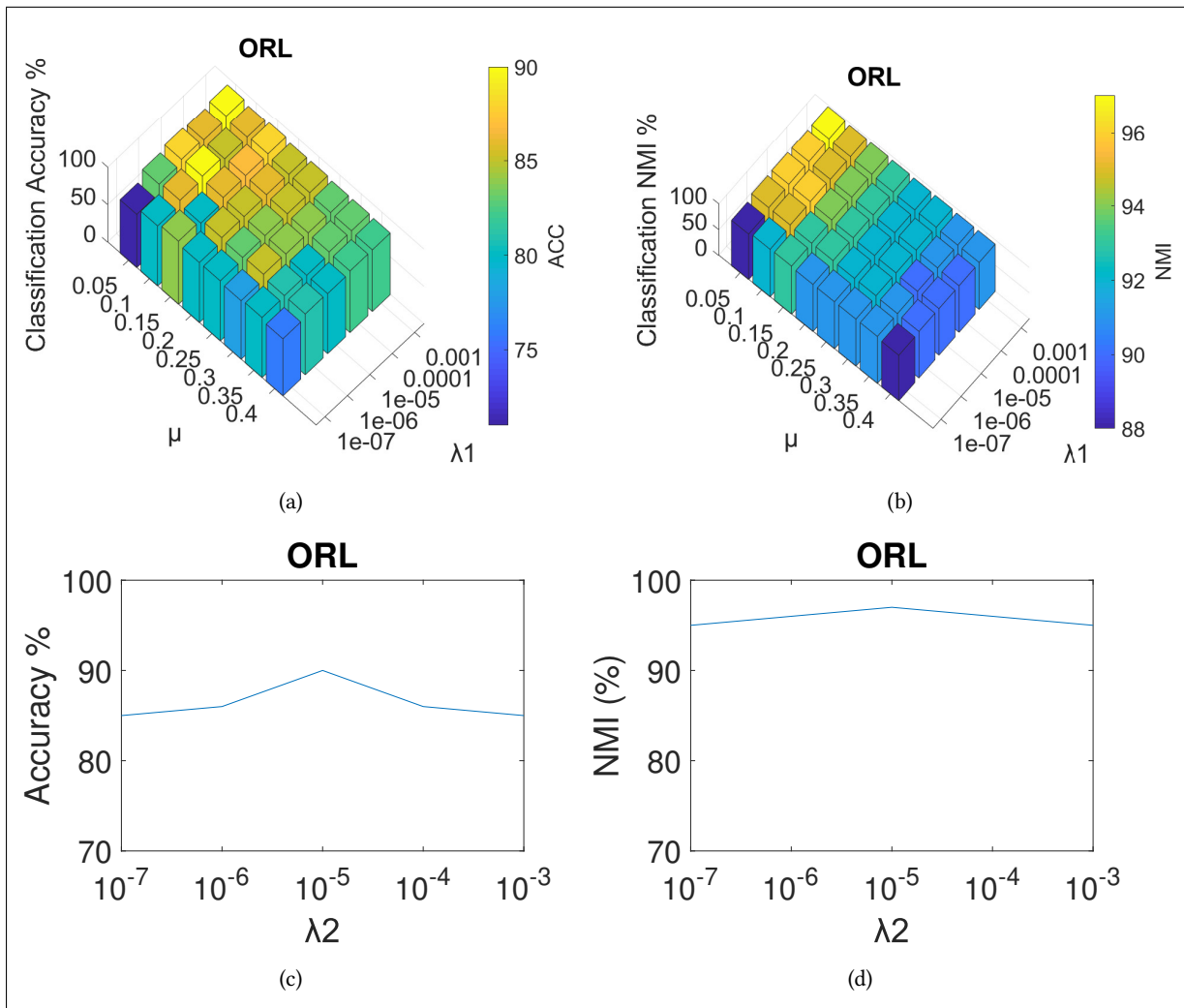


Figure 5.2: Clustering performance as a function of the balance parameters using the ORL dataset. (a) and (c) depict ACC (%). (b) and (d) depict NMI (%).

the MVCSK method. For the MSRCv1 dataset, the result of our method is slightly lower than that of some other methods, but still higher than that of the two competing methods: NESE and MVCSK, which demonstrates the good performance of our method.

On the large dataset MNIST, it can be seen from Table 5.2 that the performance obtained by our method is close to the performance of CNESE and NESE methods and is also superior to MVCSK method, which shows the efficiency of our algorithm even on large datasets. Furthermore, our method is inspired by both: NESE and MVCSK. Therefore, our work is mainly concerned with outperforming the two mentioned methods.

In addition, the result obtained for the subset of 1000 images from the MNIST dataset (denoted as MNIST-1000 in Table 5.2) shows the obvious superiority of our method. Thus, Table 5.2 shows that our approach is better than other approaches for different sizes of datasets.

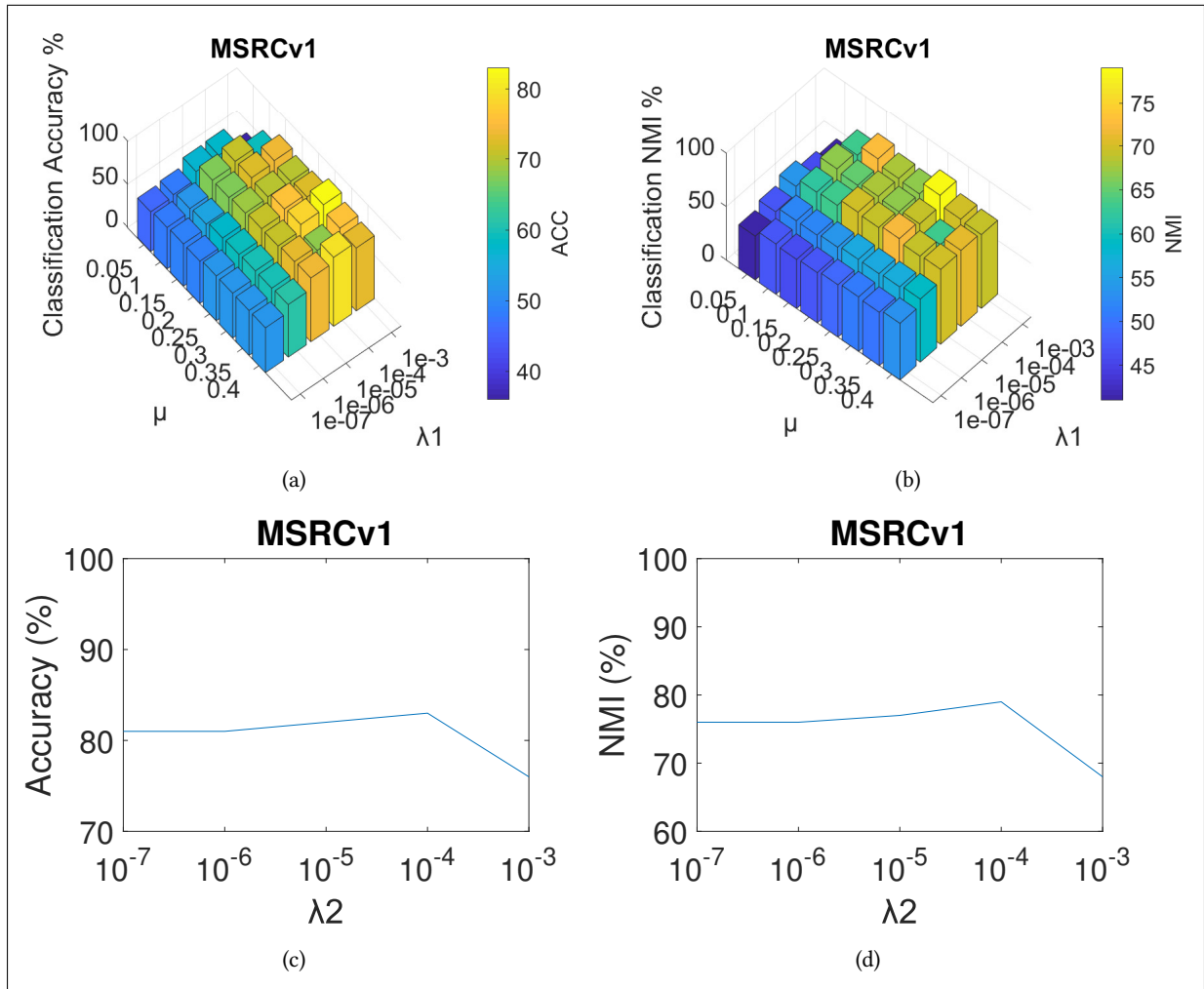


Figure 5.3: Clustering performance as a function of the balance parameters using the MSRCv1 dataset. (a) and (c) depict ACC (%). (b) and (d) depict NMI (%).

Our experimental results show that the proposed method was slightly outperformed by the NESE and CNESE methods on the MSRCv1, NUS, and MNIST datasets (only for some indicators). A plausible reason for this is the type of criterion optimized in each method. In NESE and CNESE, the individual graphs were precomputed and used as is, and the unified cluster-label matrix was constrained by the graph of each view via automatic weights. In contrast, our proposed criterion attempts to estimate a unified graph to which cluster assignments are constrained.

One of the reasons for the good performance of our method is that no post-processing step like k-means is required to obtain the clusters. Moreover, our proposed method can simultaneously produce the consensus similarity matrix across all views, the spectral projection matrix and the consistent nonnegative cluster indices of the raw data.

Therefore, from these results, we can conclude that our approach has good performance for different types and sizes of datasets.

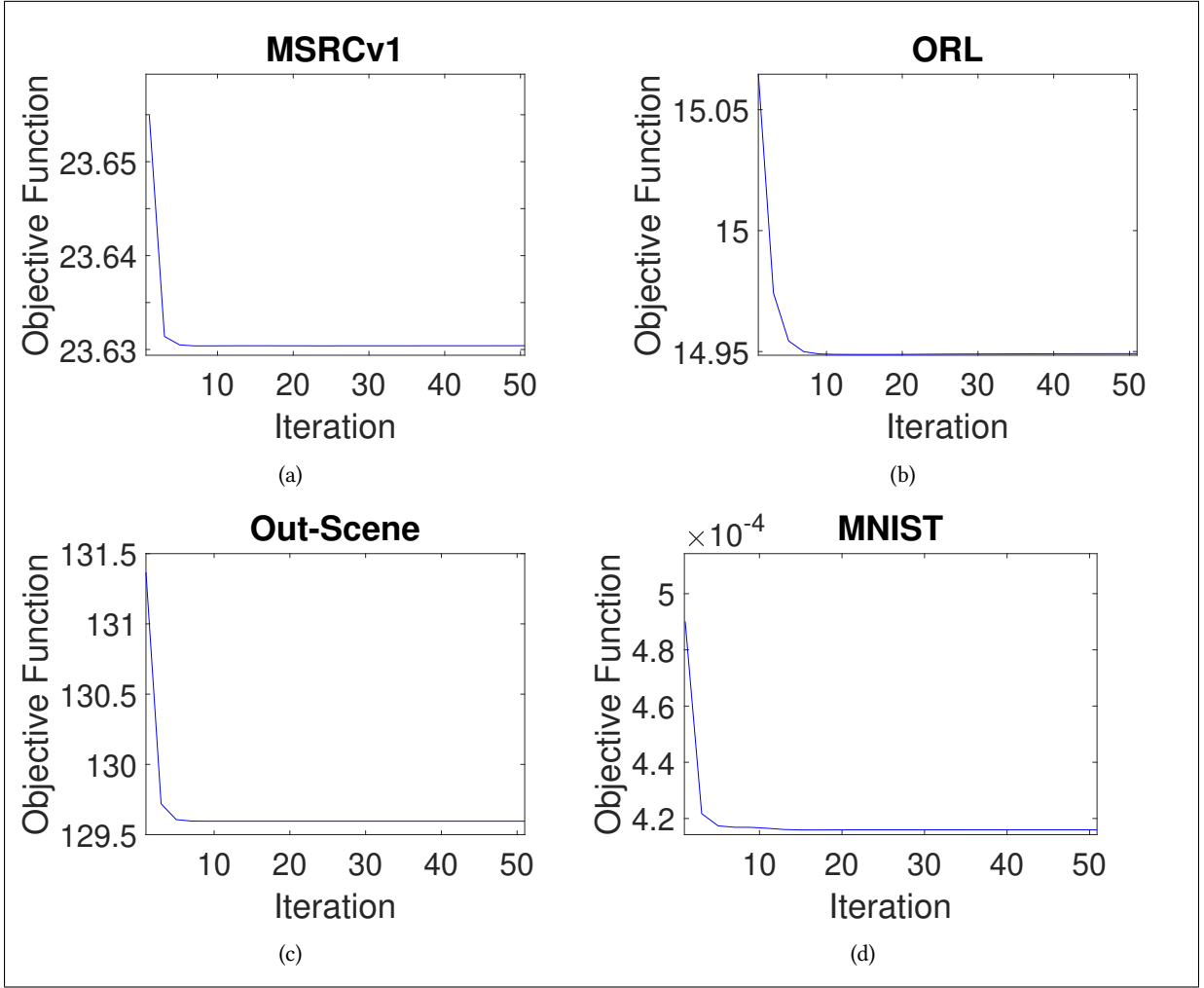


Figure 5.4: Objective function versus iteration number on four different datasets.

5.2.6 Convergence study

In this section, we examine the convergence of our proposed method. The convergence was proved theoretically in section 5.1.3. However, here we can show it empirically. To this end, we evaluate the objective function of our method at each iteration. The objective function we minimize is given by:

$$\sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})} + \mu \|\mathbf{S}\|_2^2 + \lambda_1 \text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \quad (5.28)$$

Figure 5.4 shows the evolution of the objective function as a function of the iteration number. This figure refers to four different datasets. In this figure, the horizontal axis corresponds to the number of iterations and the vertical axis corresponds to the objective value of our method.

According to this figure, the proposed method converges quickly. For the MSRCv1 and ORL datasets, convergence was achieved in less than 10 iterations. Even for the large datasets, e.g., Out-Scene and MNIST, convergence was achieved in 10 iterations.

5.2.7 Sequential estimation vs. proposed approach

To investigate the utility of the proposed simultaneous estimation of the consistent graph, spectral projection, and nonnegative cluster index matrix, we compared it to a sequential approach using the same individual terms shown in Eq. (5.8). In this sequential approach, we follow the order adopted by classic spectral clustering methods that sequentially run three stages, i.e., similarity matrix learning from data, spectral representation learning, and k-means clustering on spectral representation, respectively. In this approach, we first estimate the consistent graph matrix \mathbf{S} by minimizing the term $\sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S})} + \mu \|\mathbf{S}\|_2^2$. We then estimate the spectral representation, \mathbf{P} , from the obtained graph by minimizing $\text{Tr}(\mathbf{P}^T \mathbf{L} \mathbf{P})$. Finally, the clustering matrix \mathbf{H} is estimated from both the graph and the spectral projection using Eq. (5.12).

Table 5.3 shows a comparison between our proposed approach and the sequential approach (denoted by SA) applied to the COIL20 and Out-Scene datasets. It can be seen that the simultaneous estimation of the matrices gives better results than computing them in a sequential manner that achieves individual optimization.

Table 5.3: Method comparison on COIL20 and Out-Scene datasets.

Dataset	Method	ACC	NMI	Purity	ARI
COIL20	SA	0.65	0.81	0.68	0.60
	MKGNE	0.95	0.99	0.95	0.95
Out-Scene	SA	0.52	0.40	0.53	0.33
	MKGNE	0.65	0.54	0.65	0.43

5.2.8 Computational complexity

In this section, we study the computational complexity of the proposed method. Our algorithm consists of four main steps: updating \mathbf{H} , \mathbf{P} , \mathbf{S} , and w_v (see **Algorithm 1**). The calculation of the V kernel matrices has a computational cost of $\mathcal{O}(n^2 k)$ where k is the sum of the dimensions of the instances in the V views. Thus, $k = d^1 + d^2 + \dots + d^V$.

By inspecting the four steps of **Algorithm 1**, we can see easily that steps 2 and 3 are the most expensive ones. Indeed, step 1 consists of matrix multiplication. Moreover, step 4 contains simple matrix multiplications and additions. Therefore, we can ignore their computation cost.

To get the matrix \mathbf{P} (step 2), a matrix inversion of an $n \times n$ matrix is needed (or equivalently we should solve a linear system whose square matrix size is $n \times n$). If the orthogonalization of the matrix \mathbf{P} is invoked, one should add the associated cost which is $\mathcal{O}(n C^2)$ where C is the number

of clusters. To estimate the graph matrix \mathbf{S} (step 3), one matrix inversion of size $n \times n$ is needed. Thus, the computational cost of the third step is $\mathcal{O}(n^3)$.

Let τ be the number of iterations of the proposed iterative algorithm. The overall computational complexity of the proposed method will be $\mathcal{O}(n^2k + \tau(n^3 + nC^2 + n^3)) = \mathcal{O}(n^2k + \tau(nC^2 + n^3))$. Table 5.4 shows the computational time cost of the three main multi-view clustering methods: MVCSK, NESE, and our proposed method.

Table 5.4: Time cost (s) of MVCSK, NESE and MKGNE algorithms respectively.

Method/Dataset	COIL20	ORL	Out-Scene
MVCSK	668.79	45.67	4541.20
NESE	10.36	1.48	25.01
MKGNE	198.93	4.25	450.88

According to Table 5.4, the time required by our algorithm is longer than that required by NESE and shorter than that required by MVCSK. This can be justified by the steps required by our algorithm to compute and update the unified graph matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ and the time required to update the two matrices \mathbf{P} and $\mathbf{H} \in \mathbb{R}^{n \times C}$. In contrast, in the NESE algorithm, the single-view graphs \mathbf{S}^v are given as input to the algorithm and do not change during optimization. MVCSK takes more time than other algorithms due to the time required for eigenvalue decomposition.

5.2.9 Clustering visualization

To better understand the behavior of the proposed clustering method, we created a visual representation of the data using the t-SNE visualization technique [98]. For this purpose, we consider the ORL dataset. Figure 5.5 shows four estimated clusters. Figure 5.6 represents the t-SNE visualization of all images located in the four clusters. Each visualization corresponds to a particular individual view.

The five different colors of the dots in Figure 5.6 represent the ground truth identities of the 5 persons that appear in Figure 5.5, which represents the estimated four clusters obtained by our method. The images associated with the first cluster are depicted in the first row of Figure 5.5. These images are shown within black circles in each view. This cluster contains 11 images: 10 images for person 1 and one image for person 2. For presentation clarity, we only visualize ten images (first row in Figure 5.5). It is obvious that this estimated cluster is found correctly except for one image that corresponds to person 2, showing the high efficiency of our method.

Our method also provides a consistent spectral projection denoted by the matrix \mathbf{P} . In Figure 5.6(f), we provided the t-SNE visualization of the same 40 images (shown above) in the unified spectral



Figure 5.5: Four clusters provided by our approach for the ORL dataset.

space. This visualization explains better why the first cluster contained one image from person 2. The first cluster contains eleven images presented by 10 red dots (the first person) and one image presented by a lime dot (which refers to the second person). We can also observe that the red dot (images of the first cluster) have better compact distribution in the spectral representation.

5.3 Conclusion

In this chapter, we present a new approach for multi-view clustering. This approach is characterized by its ability to simultaneously estimate the unified graph similarity matrix, the unified spectral embedding, and the nonnegative cluster index matrix. Moreover, the proposed approach can automatically estimate the weight of each view without using additional parameters. Experimental results performed on datasets of different types and sizes show the efficiency of our approach compared to other methods.

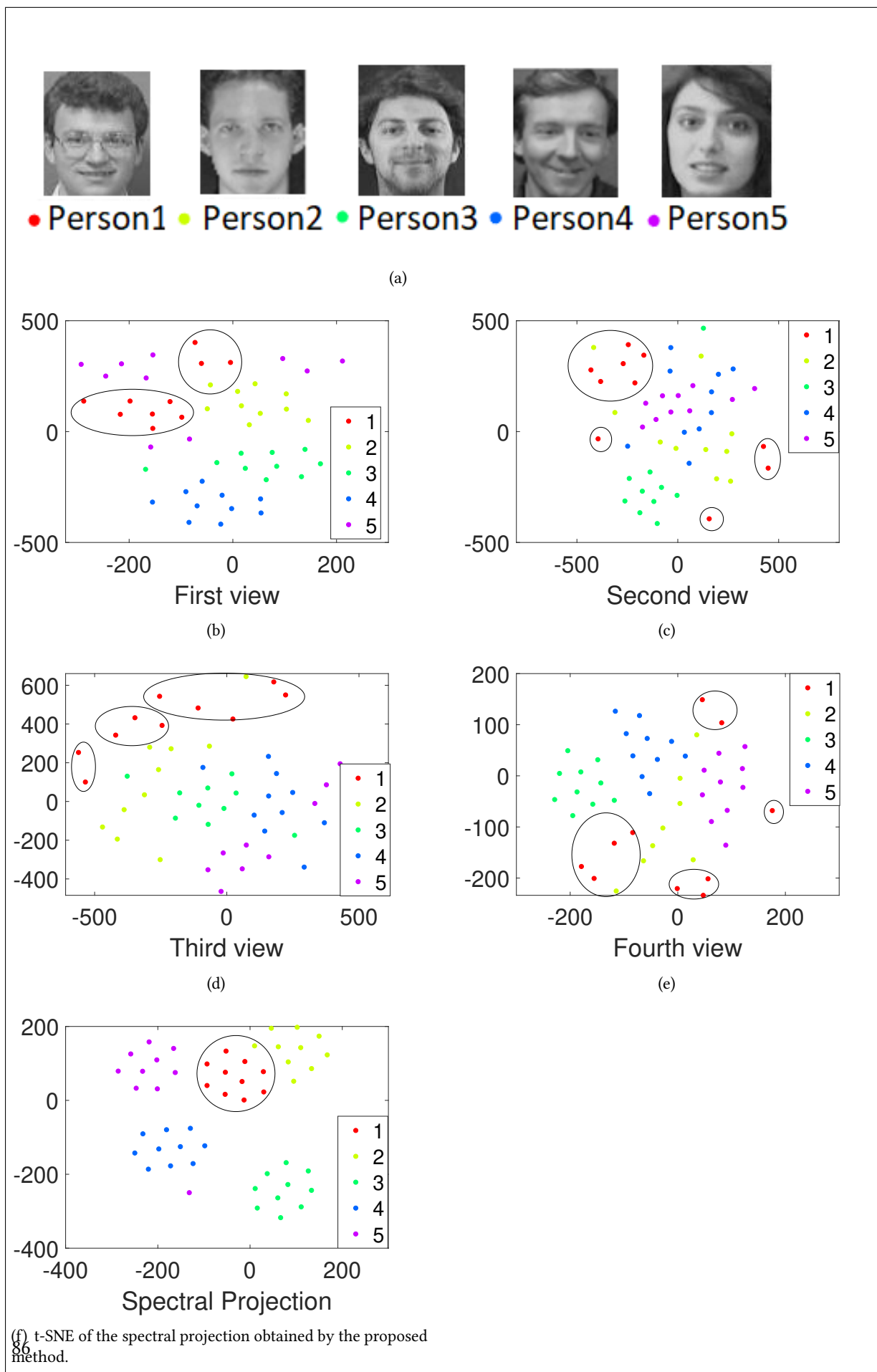


Figure 5.6: t-SNE of the original features and the spectral projection of ORL dataset.

Consensus graph and spectral representation for one-step multi-view kernel based clustering

In this chapter, we present a novel method similar to the method presented in Chapter 5. Same as the MKGNE method, it can address some of the limitations of previous multi-view clustering methods. *The first difference between this novel approach and the MKGNE method, is the second term in the objective function. This term was a smoothness constraint of the spectral projection matrix P over the unified graph in the case of the MKGNE method. However, in our novel proposed approach, it represents the smoothness constraint of the nonnegative embedding matrix H (soft clustering assignments) over the graphs. The second difference between the MKGNE method and our novel model is the orthogonality constraint over the nonnegative embedding matrix H .* This approach, called Multi-view Clustering via Consensus Graph Learning and Nonnegative Embedding (MVCGE), provides a consistent nonnegative embedding matrix to determine the final cluster assignment. It estimates the clusters of the data directly without any additional post-processing and enforces the cluster index matrix to be a kind of convolution of a unified spectral representation over a consistent graph. This method can overcome some of the drawbacks of other approaches since it provides simultaneously the consistent similarity matrix, the nonnegative cluster index matrix, and the unified spectral projection matrix across all views. Moreover, the proposed approach automatically calculates the weight of each view without using any additional parameters. It takes as input the different kernel matrices corresponding to the different views. The proposed learning model integrates two interesting

constraints: (i) the cluster indices should be as smooth as possible over the consensus graph; and (ii) the cluster indices are set to be as close as possible to the graph convolution of the consensus representation.

Similar to the MKGNE method, the novel approach presented in this chapter combines the advantages of graph-based methods and matrix factorization methods, such as MVCSK in [24] and NESE in [11]. Since our method combines the advantages of NESE and MVCSK, the main goal of our study is to outperform the previous two methods, as it will be shown in the extensive experimentation part. The contributions of this chapter are summarized below.

1. Unlike other approaches based on multi-view learning, our method can simultaneously provide the consensus similarity matrix, the nonnegative index cluster matrix, the spectral projection matrix, and the weight of each view automatically.
2. It generates the final clustering assignment directly without any post-processing step. Our method inherits the advantages of matrix factorization methods and graph-based methods.
3. The proposed model successfully finds nonlinear interactions between different views. This model is able to compute the exact graph considering the underlying correlations from numerous views by using a kernel representation of each view.
4. The cluster index matrix, which is the consequence of the convolution of the coherent spectral projection matrix over the coherent graph, is learned as part of the proposed learning technique.
5. It has been validated on real and synthetic datasets. This validation shows that this approach can give better results compared to state-of-the-art clustering methods.

6.1 Proposed Approach

We introduce a new approach called Multi-View Clustering via Consensus Graph Learning and Nonnegative Embedding (MVCGE), which combines the advantages of graph learning methods and matrix factorization methods. MVCGE achieves the clustering results without any additional step. Similar to the MKGNE method, and inspired by the MVCSK method the first term of our method is given below:

$$\min_{\mathbf{S}} \sum_{v=1}^V w_v Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \alpha \|\mathbf{S}\|_2^2 \quad s.t. \quad \mathbf{S} \geq 0. \quad (6.1)$$

The clustering result is obtained from the nonnegative embedding matrix \mathbf{H} , which provides the cluster indices by taking the index of the highest element in the row vector $\mathbf{H}_{i*} \in \mathbb{R}^K$. Since the matrix \mathbf{H} is used for the final cluster assignment, it is important to use a smoothing term for this matrix so that it is more coherent with the graph entries. The smoothing term ensures that two data points \mathbf{x}_i^v and \mathbf{x}_j^v that are similar (i.e., the value of the corresponding value in the similarity matrix S_{ij} is large) are necessarily in the same cluster (i.e., the corresponding cluster index \mathbf{H}_{i*} and \mathbf{H}_{j*} are close). Therefore, the second term of our criterion is given by:

$$\min_{\mathbf{H}} \frac{1}{2} \sum_i \sum_j \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2 S_{ij} = \min_{\mathbf{H}} \text{Tr}(\mathbf{H}^T \mathbf{L} \mathbf{H}), \quad (6.2)$$

where $\mathbf{L} = \mathbf{D} - \mathbf{S} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the consistent graph matrix, and \mathbf{D} is a diagonal matrix whose elements are given by: $D_{ii} = \sum_{j=1}^n \frac{S_{ij} + S_{ji}}{2}$. The third term of our proposed method states that the cluster index of the i -th instance (the row vector $\mathbf{H}_{i,*}$) is set to the convolution of the spectral representation \mathbf{P} with the i -th row of the graph matrix $\mathbf{S}_{i,*}$. This approach has two main advantages. First, the clustering is performed in a single step. Second, the clustering uses the consolidated spectral representation of the neighbors obtained in the consensus graph.

Moreover, inspired by the principle of data convolution, the nonnegative embedding matrix used to obtain the final clustering assignment will be equal to " $\mathbf{H} = \max(\mathbf{S}\mathbf{P}, 0)$ ". This means that the nonnegative matrix is the result of the convolution of the spectral data representation with the graph. The third term of our criterion binds the cluster index label to the consensus spectral representation. Therefore, the cluster index matrix should satisfy the following condition:

$$\min_{\mathbf{H} \geq 0, \mathbf{P}^T \mathbf{P} = \mathbf{I}} \|\mathbf{H} - \mathbf{S}\mathbf{P}\|_F^2, \quad (6.3)$$

where the matrix $\mathbf{P} \in \mathbb{R}^{n \times C}$ is a consensus data representation. In our work, it is initialized to a unified spectral representation of the data.

Since the matrix \mathbf{P} is orthogonal, Eq. (6.3) can take another form to illustrate the factorization of the graph matrix \mathbf{S} using the nonnegative embedding matrix \mathbf{H} and the spectral projection matrix \mathbf{P} . It can be written as:

$$\min_{\mathbf{H}, \mathbf{P}} \|\mathbf{S} - \mathbf{H}\mathbf{P}^T\|_F^2 \quad s.t. \quad \mathbf{H} \geq 0, \quad \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (6.4)$$

Our final objective function is obtained by adding the three terms from equations (6.1), (6.2), and (6.4).

$$\begin{aligned} \min_{\mathbf{S}, \mathbf{P}, \mathbf{H}} \sum_{v=1}^V w_v Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \alpha \|\mathbf{S}\|_2^2 + \lambda_1 Tr(\mathbf{H}^T \mathbf{L} \mathbf{H}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \\ s.t. \mathbf{S} \geq 0, \mathbf{P}^T \mathbf{P} = \mathbf{I}, \mathbf{H}^T \mathbf{H} = \mathbf{I}, \mathbf{H} \geq 0, \end{aligned} \quad (6.5)$$

where α , λ_1 and λ_2 are three regularization parameters.

Optimization:

We use an iterative update procedure to solve our objective function. In MVCGE, three matrices are unknown: \mathbf{S} , \mathbf{H} , and \mathbf{P} . An alternating optimization scheme is used for the optimization procedure. We proceed as follows:

Step 1: Fix all, estimate H: The problem (6.5) is:

$$\min_{\mathbf{H}} Tr(\mathbf{H}^T \mathbf{L} \mathbf{H}) + \frac{\lambda_2}{\lambda_1} \|\mathbf{S} \mathbf{P} - \mathbf{H}\|_2^2 \quad s.t. \mathbf{H}^T \mathbf{H} = \mathbf{I}, \mathbf{H} \geq 0. \quad (6.6)$$

Vanishing the derivative of (6.6) w.r.t. \mathbf{H} yields:

$$\mathbf{H} = \left(\mathbf{L} + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right)^{-1} \frac{\lambda_2}{\lambda_1} \mathbf{S} \mathbf{P}. \quad (6.7)$$

To satisfy the orthogonality and non-negativity constraints, an orthogonalization step is first applied to the obtained \mathbf{H} , then the negative values of \mathbf{H} are set to zero.

Step 2: Fix all, estimate P: The problem (6.5) becomes:

$$\min_{\mathbf{H}} \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2. \quad (6.8)$$

Since \mathbf{P} is orthogonal, i.e., $\mathbf{P}^T \mathbf{P} = \mathbf{I}$, \mathbf{P} is obtained by performing the singular value decomposition of $\mathbf{S}^T \mathbf{H}$. Let $\mathbf{U} \Sigma \mathbf{V}^T = \text{SVD}(\mathbf{S}^T \mathbf{H})$, then the solution of (6.8) is given by:

$$\mathbf{P} = \mathbf{U} \mathbf{V}^T \quad \text{with} \quad \mathbf{U} \Sigma \mathbf{V}^T = \text{SVD}(\mathbf{S}^T \mathbf{H}). \quad (6.9)$$

Step 3: Fix all, estimate S:

If we fix \mathbf{H} and \mathbf{P} , we need to solve the following problem:

$$\min_{\mathbf{S}} \sum_{v=1}^V w_v Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \alpha \|\mathbf{S}\|_2^2 + \lambda_1 Tr(\mathbf{H}^T \mathbf{L} \mathbf{H}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \quad s.t. \mathbf{S} \geq 0. \quad (6.10)$$

After the spectral clustering analysis, we have the known identity:

$$Tr(\mathbf{H}^T \mathbf{L} \mathbf{H}) = \frac{1}{2} \sum_i \sum_j \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2 S_{ij} = Tr(\mathbf{Q} \mathbf{S}), \quad (6.11)$$

where \mathbf{H}_{i*} is the i -th row of \mathbf{H} . The symmetric matrix \mathbf{Q} denotes the pairwise distance associated with the rows of the matrix \mathbf{H} . It is given by $Q_{ij} = \frac{1}{2} \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2$. Substituting Eq. (6.11) into Eq. (6.10), the latter becomes:

$$\min_{\mathbf{S}} \sum_{v=1}^V w_v Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S} + \mathbf{S}^T \mathbf{K}^v \mathbf{S}) + \alpha \|\mathbf{S}\|_2^2 + \lambda_1 Tr(\mathbf{Q} \mathbf{S}) + \lambda_2 \|\mathbf{S} - \mathbf{H} \mathbf{P}^T\|_2^2 \quad s.t. \quad \mathbf{S} \geq 0. \quad (6.12)$$

By making the derivative of Eq. (6.12) w.r.t. \mathbf{S} vanish, we obtain \mathbf{S} as ($ReLU()$ is the Rectified Linear Unit function):

$$\mathbf{s} = ReLU \left\{ \left(\sum_{v=1}^V w_v \mathbf{K}^v + (\alpha + \lambda_2) \mathbf{I} \right)^{-1} \left(\sum_{v=1}^V w_v \mathbf{K}^v + \lambda_2 \mathbf{H} \mathbf{P}^T - \frac{1}{2} \lambda_1 \mathbf{Q} \right) \right\}. \quad (6.13)$$

Step 4: Fix \mathbf{H} , \mathbf{P} , and \mathbf{S} , and update w_v ($v = 1, \dots, V$) using Eq. (5.2).

The main steps of the proposed approach ‘‘Multi-view Clustering via Consensus Graph Learning and Nonnegative Embedding’’ (MVCGE) are summarized in Algorithm 1.

Algorithm 1 MVCGE

Input: Data samples in V views $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$.
The graph matrices \mathbf{S}^v , $v = 1, \dots, V$.
The spectral embedding matrices \mathbf{P}^v , $v = 1, \dots, V$.
Parameters α , λ_1 , λ_2 .

Output: The consensus graph matrix \mathbf{S} .
The consensus spectral representation matrix \mathbf{P} .
The cluster index matrix (nonnegative embedding matrix) \mathbf{H} .

Initialization:

The weight of each view $w_v = \frac{1}{V}$.
Compute the kernel matrix \mathbf{K}^v for each view.
Initialize \mathbf{S} and \mathbf{P} by taking the average of the matrices \mathbf{S}^v and \mathbf{P}^v .

Repeat

Update \mathbf{H} using Eq. (6.7).
Update \mathbf{P} using Eq. (6.9).
Update \mathbf{S} using Eq. (6.13).
Update w_v using Eq. (5.2).

Until convergence

To initialize the two matrices \mathbf{S} and \mathbf{P} , the efficient method used in [93] is used. This method finds the similarity matrix and the corresponding spectral projection matrix for each view. To obtain the initial unified matrix, the average of all the individual matrices is used.

6.2 Performance Evaluation

6.2.1 Datasets

The effectiveness of the proposed approach is evaluated using eight real datasets and three synthetic datasets. The datasets used to test this method are: COIL20, ORL, Out-Scene, BBCSport, MSRCv1, Extended-Yale, MNIST, MNIST-1000, Tetra, Hepta, and Chainlink.

6.2.2 Experimental setup

Several competing methods are used for comparison: 1) Co-training approach for multi-view Spectral Clustering (CotSC) [34], 2) Co-regularized approach for multi-view Spectral Clustering (CorSC) [35], 3) Multi-view Learning Clustering with Adaptive Neighbors (MLAN) [44], 4) Self-weighted Multi-view Clustering with multiple graphs (SwMC) [95], 5) Affinity Aggregation for Spectral Clustering (AASC) [96], 6) Graph Learning for Multi-View clustering (MVGL) [15], 7) Parameter-free Auto-weighted Multiple Graph Learning (AMGL) [13], 8) Multi-view clustering via Adaptively Weighted Procrustes (AWP) [20], 9) Auto-weighted Multi-View Clustering via Kernelized graph learning (MVCSK) [24], 10) Multi-view spectral clustering via integrating Nonnegative Embedding and Spectral Embedding (NESE) [11], 11) Sparse Multi-view Spectral Clustering (S-MVSC) [77], 12) Consistency-aware and Inconsistency-aware Graph-based Multi-View Clustering (CI-GMVC) [78], 13) Multi-View Clustering in Latent Embedding Space (MCLES) [67] and 14) multi-view spectral clustering via Constrained Nonnegative Embedding (CNESE) [97]. We also report the Spectral Clustering best view result (SC) [33].

The clustering performance of the proposed approach is compared with other methods, A Gaussian kernel function is used to construct the kernel matrix of each view. To initialize our algorithm, we use the same method as in [11]. First, the similarity matrix of each view is computed. Then, the corresponding spectral projection matrices are computed, and the final unified similarity matrix and spectral projection matrix are the average of the corresponding matrix of all views. In this way, we obtain the initial values of the matrices \mathbf{S} and \mathbf{P} .

In our method, three parameters are used: α , λ_1 and λ_2 . The values of α are in the range [0.005 0.9], the values of the parameter λ_1 vary over the set $\{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4},$

10^{-3} } and the values of the parameter λ_2 vary over the set $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

To compare our method with other methods, we use four clustering performance metrics: Accuracy (ACC), Normalized Mutual Information (NMI), Purity, and Adjusted Rand Index (ARI).

6.2.3 Experimental results

Our algorithm is tested on real and synthetic datasets. Table 6.1 shows the results obtained by MVCGE and some other methods on the datasets: ORL, Out-Scene, and Coil20. In this table, the highest scores are marked in bold. The proposed method MVCGE was superior on these datasets. For some competing methods listed in Table 6.1, the corresponding method is repeated in multiple trials, and then a standard deviation for each indicator is given in parentheses. From this table, we can see that our method and methods MVCSK, NESE S-MVSC, CI-GMVC, MCLES and CNESE perform best, so we can adopt them to test the other datasets.

Table 6.2 shows a comparison between our method and the aforementioned methods for the BBCSport, MSRCv1, Extended-Yale, MNIST and MNIST-1000 datasets. For the MNIST dataset, which is a large image dataset (i.e., the number of samples is equal to 10000), each image has two deep descriptors, which means that the data already has some nonlinearity. Then, the use of the large kernel matrices can be skipped. Therefore, the criterion of our method reduces to the last two terms, where we only update the spectral projection matrix and the nonnegative embedding matrix.

Our method is applied to the synthetic datasets: Tetra, Chainlink, and Hepta. The results are presented in Table 6.3.

6.2.4 Ablation study

Our proposed criterion (6.5) contains three main terms: the graph construction and its regularization, the smoothness term and the convolution term. To illustrate the relevance of the proposed criterion and its terms, we generate four different models with different combinations. These four different variants of MVCGE are: MVCGE-G, MVCGE-S, MVCGE-C and MVCGE-SC. (1) No graph regularization term in the global objective function (6.5) (i.e., α is set to zero), and we call the obtained method MVCGE-G, which means that only the smoothness and convolution terms in MVCGE are used, (2) No smoothness constraint (λ_1 is set to zero), and we call the obtained

Table 6.1: Clustering performance on the ORL, Outdoor-Scene and Coil20 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
ORL	SC-Best [33]	0.66 (± 0.02)	0.76 (± 0.02)	0.71 (± 0.02)	0.67 (± 0.01)
	AWP [20]	0.80 (± 0.00)	0.91 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MLAN [44]	0.78 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.67 (± 0.00)
	SwMC [95]	0.77 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)	0.62 (± 0.00)
	AMGL [13]	0.75 (± 0.02)	0.90 (± 0.02)	0.82 (± 0.02)	0.63 (± 0.09)
	AASC [96]	0.82 (± 0.02)	0.91 (± 0.01)	0.85 (± 0.01)	0.76 (± 0.02)
	MVGL [15]	0.75 (± 0.00)	0.88 (± 0.00)	0.80 (± 0.00)	0.55 (± 0.00)
	CorSC [35]	0.77 (± 0.03)	0.90 (± 0.01)	0.82 (± 0.03)	0.72 (± 0.04)
	CotSC [34]	0.75 (± 0.04)	0.87 (± 0.01)	0.78 (± 0.03)	0.67 (± 0.03)
	NESE [11]	0.82 (± 0.00)	0.91 (± 0.00)	0.85 (± 0.00)	0.75 (± 0.00)
	MVCSK [24]	0.85 (± 0.02)	0.94 (± 0.01)	0.88 (± 0.02)	0.81 (± 0.02)
	S-MVSC [77]	0.80 (± 0.02)	0.93 (± 0.01)	0.82 (± 0.02)	0.89 (± 0.01)
	CI-GMVC [78]	0.81 (± 0.00)	0.92 (± 0.00)	0.85 (± 0.00)	0.74 (± 0.00)
	MCLES [67]	0.84 (± 0.00)	0.94 (± 0.00)	0.88 (± 0.00)	0.79 (± 0.00)
CNESE [97]	0.87 (± 0.00)	0.95 (± 0.00)	0.89 (± 0.00)	0.84 (± 0.00)	
MVCGE	0.93 (± 0.00)	0.97 (± 0.00)	0.95 (± 0.00)	0.92 (± 0.00)	
Out-Scene	SC-best [33]	0.47 (± 0.01)	0.39 (± 0.01)	0.57 (± 0.01)	0.34 (± 0.01)
	AWP [20]	0.65 (± 0.00)	0.51 (± 0.00)	0.65 (± 0.00)	0.42 (± 0.00)
	MLAN [44]	0.55 (± 0.02)	0.47 (± 0.01)	0.55 (± 0.02)	0.33 (± 0.03)
	SwMC [95]	0.50 (± 0.00)	0.47 (± 0.00)	0.50 (± 0.00)	0.38 (± 0.00)
	AMGL [13]	0.51 (± 0.05)	0.45 (± 0.03)	0.52 (± 0.04)	0.34 (± 0.05)
	AASC [96]	0.60 (± 0.00)	0.48 (± 0.00)	0.60 (± 0.00)	0.35 (± 0.00)
	MVGL [15]	0.42 (± 0.00)	0.31 (± 0.00)	0.43 (± 0.00)	0.16 (± 0.00)
	CorSC [35]	0.51 (± 0.04)	0.39 (± 0.03)	0.52 (± 0.03)	0.31 (± 0.02)
	CotSC [34]	0.38 (± 0.02)	0.22 (± 0.01)	0.39 (± 0.02)	0.16 (± 0.01)
	NESE [11]	0.63 (± 0.00)	0.53 (± 0.00)	0.66 (± 0.00)	0.46 (± 0.00)
	MVCSK [24]	0.65 (± 0.01)	0.52 (± 0.00)	0.65 (± 0.01)	0.42 (± 0.00)
	S-MVSC [77]	0.48 (± 0.01)	0.54 (± 0.02)	0.65 (± 0.01)	0.46 (± 0.04)
	CI-GMVC [78]	0.35 (± 0.01)	0.31 (± 0.00)	0.35 (± 0.01)	0.19 (± 0.00)
	MCLES [67]	0.65 (± 0.00)	0.53 (± 0.00)	0.67 (± 0.00)	0.46 (± 0.00)
CNESE [97]	0.66 (± 0.00)	0.55 (± 0.00)	0.67 (± 0.00)	0.47 (± 0.00)	
MVCGE	0.70 (± 0.00)	0.55 (± 0.00)	0.70 (± 0.00)	0.47 (± 0.00)	
COIL20	SC-Best [33]	0.73 (± 0.01)	0.82 (± 0.01)	0.75 (± 0.01)	0.68 (± 0.02)
	AWP [20]	0.68 (± 0.00)	0.87 (± 0.00)	0.75 (± 0.00)	0.71 (± 0.00)
	MLAN [44]	0.84 (± 0.00)	0.92 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)
	SwMC [95]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.84 (± 0.00)
	AMGL [13]	0.80 (± 0.04)	0.91 (± 0.02)	0.85 (± 0.03)	0.74 (± 0.07)
	AASC [96]	0.79 (± 0.00)	0.89 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MVGL [15]	0.78 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)	0.75 (± 0.00)
	CorSC [35]	0.68 (± 0.04)	0.78 (± 0.02)	0.70 (± 0.03)	0.62 (± 0.03)
	CotSC [34]	0.70 (± 0.03)	0.80 (± 0.02)	0.72 (± 0.03)	0.65 (± 0.03)
	NESE [11]	0.77 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.69 (± 0.00)
	MVCSK [24]	0.65 (± 0.04)	0.80 (± 0.02)	0.70 (± 0.03)	0.61 (± 0.05)
	S-MVSC [77]	0.62 (± 0.01)	0.86 (± 0.02)	0.77 (± 0.02)	0.97 (± 0.02)
	CI-GMVC [78]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)
	MCLES [67]	0.79 (± 0.00)	0.88 (± 0.00)	0.83 (± 0.00)	0.75 (± 0.00)
CNESE [97]	0.82 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.78 (± 0.00)	
MVCGE	1.00 (± 0.00)	1.00 (± 0.00)	1.00 (± 0.00)	1.00 (± 0.00)	

Table 6.2: Clustering performance on the BBCSport, MSRCv1, Extended-Yale, MNIST and MNIST-1000 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
BBCSport	MVCSK [24]	0.90 (\pm 0.07)	0.82 (\pm 0.02)	0.90 (\pm 0.02)	0.85 (\pm 0.07)
	NESE [11]	0.72 (\pm 0.00)	0.69 (\pm 0.00)	0.75 (\pm 0.00)	0.60 (\pm 0.00)
	S-MVSC [77]	0.58 (\pm 0.07)	0.67 (\pm 0.01)	0.73 (\pm 0.02)	0.83 (\pm 0.04)
	CI-GMVC [78]	0.61 (\pm 0.00)	0.46 (\pm 0.00)	0.63 (\pm 0.00)	0.36 (\pm 0.00)
	MCLES [67]	0.88 (\pm 0.00)	0.80 (\pm 0.00)	0.88 (\pm 0.00)	0.83 (\pm 0.00)
	CNESE [97]	0.72 (\pm 0.00)	0.68 (\pm 0.00)	0.76 (\pm 0.00)	0.60 (\pm 0.00)
	MVCGE	0.98 (\pm 0.00)	0.94 (\pm 0.00)	0.98 (\pm 0.00)	0.95 (\pm 0.00)
MSRCv1	MVCSK [24]	0.70 (\pm 0.02)	0.59 (\pm 0.03)	0.70 (\pm 0.02)	0.50 (\pm 0.04)
	NESE [11]	0.77 (\pm 0.00)	0.72 (\pm 0.00)	0.80 (\pm 0.00)	0.64 (\pm 0.00)
	S-MVSC [77]	0.60 (\pm 0.00)	0.69 (\pm 0.02)	0.74 (\pm 0.02)	0.79 (\pm 0.01)
	CI-GMVC [78]	0.74 (\pm 0.00)	0.72 (\pm 0.00)	0.77 (\pm 0.00)	0.59 (\pm 0.00)
	MCLES [67]	0.90 (\pm 0.01)	0.83 (\pm 0.02)	0.90 (\pm 0.01)	0.77 (\pm 0.00)
	CNESE [97]	0.86 (\pm 0.00)	0.76 (\pm 0.00)	0.86 (\pm 0.00)	0.72 (\pm 0.00)
	MVCGE	0.93 (\pm 0.00)	0.87 (\pm 0.00)	0.93 (\pm 0.00)	0.85 (\pm 0.00)
Extended-Yale	MVCSK [24]	0.33 (\pm 0.00)	0.42 (\pm 0.00)	0.34 (\pm 0.00)	0.18 (\pm 0.00)
	NESE [11]	0.43 (\pm 0.00)	0.58 (\pm 0.00)	0.47 (\pm 0.00)	0.25 (\pm 0.00)
	S-MVSC [77]	0.48 (\pm 0.03)	0.61 (\pm 0.01)	0.60 (\pm 0.01)	0.36 (\pm 0.05)
	CI-GMVC [78]	0.32 (\pm 0.00)	0.34 (\pm 0.00)	0.35 (\pm 0.00)	0.02 (\pm 0.00)
	MCLES [67]	0.48 (\pm 0.03)	0.48 (\pm 0.00)	0.48 (\pm 0.01)	0.10 (\pm 0.05)
	CNESE [97]	0.60 (\pm 0.00)	0.75 (\pm 0.00)	0.60 (\pm 0.00)	0.51 (\pm 0.00)
	MVCGE	0.88 (\pm 0.00)	0.86 (\pm 0.00)	0.88 (\pm 0.00)	0.77 (\pm 0.00)
MNIST	MVCSK [24]	0.49 (\pm 0.00)	0.41 (\pm 0.00)	0.50 (\pm 0.00)	0.29 (\pm 0.00)
	NESE [11]	0.81 (\pm 0.00)	0.83 (\pm 0.00)	0.85 (\pm 0.00)	0.76 (\pm 0.00)
	S-MVSC [77]	0.77 (\pm 0.01)	0.81 (\pm 0.01)	0.81 (\pm 0.02)	0.76 (\pm 0.07)
	CI-GMVC [78]	0.66 (\pm 0.00)	0.71 (\pm 0.00)	0.71 (\pm 0.00)	0.51 (\pm 0.00)
	MCLES [67]	0.80 (\pm 0.00)	0.83 (\pm 0.00)	0.85 (\pm 0.00)	0.77 (\pm 0.00)
	CNESE [97]	0.81 (\pm 0.00)	0.83 (\pm 0.00)	0.86 (\pm 0.00)	0.78 (\pm 0.00)
	MVCGE	0.81 (\pm 0.00)	0.83 (\pm 0.00)	0.85 (\pm 0.00)	0.77 (\pm 0.00)
MNIST-1000	MVCSK [24]	0.70 (\pm 0.00)	0.61 (\pm 0.00)	0.70 (\pm 0.00)	0.52 (\pm 0.00)
	NESE [11]	0.78 (\pm 0.00)	0.79 (\pm 0.00)	0.83 (\pm 0.00)	0.71 (\pm 0.00)
	S-MVSC [77]	0.66 (\pm 0.02)	0.76 (\pm 0.01)	0.76 (\pm 0.00)	0.77 (\pm 0.05)
	CI-GMVC [78]	0.65 (\pm 0.00)	0.71 (\pm 0.00)	0.73 (\pm 0.00)	0.50 (\pm 0.00)
	MCLES [67]	0.73 (\pm 0.02)	0.72 (\pm 0.01)	0.77 (\pm 0.02)	0.58 (\pm 0.04)
	CNESE [97]	0.77 (\pm 0.00)	0.77 (\pm 0.00)	0.81 (\pm 0.00)	0.68 (\pm 0.00)
	MVCGE	0.86 (\pm 0.00)	0.83 (\pm 0.00)	0.86 (\pm 0.00)	0.78 (\pm 0.00)

method MVCGE-S, (3) No convolution term (λ_2 is set to zero), and we call the obtained method MVCGE-C, and (4) No smoothness and no convolution terms (λ_1 and λ_2 are set to zero). This method is called MVCGE-SC because it is reduced to a consistent graph construction followed by a spectral clustering step. The results obtained with MVCGE-G, MVCGE-S, MVCGE-C, MVCGE-SC and MVCGE are summarized in Table 6.4. We used three datasets: ORL, MSRCv1 and Tetra. From the results in Table 6.4, we can see that the regularization of the graph is indeed crucial, since the last two terms depend on this graph. For the ORL and Tetra datasets, it can be seen from the table that the smoothness term has a larger impact on the clustering results. However, for the MSRCv1 dataset, the convolution term is more important than the smoothness term. This is normal and is due to the different types of datasets used in this work. The results obtained with MVCGE-SC show the importance of the last two terms in the objective function for all datasets. All these

Table 6.3: Clustering performance on the three synthetic datasets.

Dataset	Method	ACC	NMI	Purity	ARI
Tetra	NESE [11]	0.64	0.75	0.75	0.63
	MVCSK [24]	0.97	0.93	0.97	0.92
	S-MVSC [77]	0.70	0.50	0.44	0.70
	CI-GMVC [78]	0.63	0.52	0.67	0.43
	MCLES [67]	0.85	0.88	0.89	0.80
	CNESE [97]	0.66	0.62	0.75	0.54
	MVCGE	1.00	1.00	1.00	1.00
Hepta	NESE [11]	0.81	0.79	0.85	0.73
	MVCSK [24]	0.89	0.85	0.89	0.80
	S-MVSC [77]	0.66	0.63	0.47	0.70
	CI-GMVC [78]	0.77	0.76	0.81	0.68
	MCLES [67]	0.87	0.82	0.84	0.80
	CNESE [97]	0.78	0.70	0.79	0.63
	MVCGE	0.92	0.85	0.92	0.83
Chainlink	NESE [11]	0.93	0.69	0.93	0.73
	MVCSK [24]	0.63	0.05	0.63	0.07
	S-MVSC [77]	0.67	0.14	0.78	0.12
	CI-GMVC [78]	0.55	0.01	0.55	0.01
	MCLES [67]	0.90	0.72	0.86	0.76
	CNESE [97]	0.95	0.70	0.95	0.78
	MVCGE	0.96	0.78	0.96	0.85

results indicate that the inclusion of all terms in the objective function contributed to the good clustering performance of our proposed method.

Table 6.4: Ablation study with different models.

Dataset	Variant	ACC	NMI	Purity	ARI
ORL	MVCGE-G	0.46	0.66	0.48	0.27
	MVCGE-S	0.75	0.88	0.76	0.72
	MVCGE-C	0.86	0.94	0.88	0.81
	MVCGE-SC	0.69	0.86	0.75	0.57
	MVCGE	0.93	0.97	0.95	0.92
MSRCv1	MVCGE-G	0.68	0.57	0.68	0.45
	MVCGE-S	0.72	0.63	0.74	0.54
	MVCGE-C	0.70	0.60	0.70	0.50
	MVCGE-SC	0.59	0.54	0.61	0.37
	MVCGE	0.93	0.87	0.93	0.85
Tetra	MVCGE-G	0.70	0.59	0.72	0.55
	MVCGE-S	0.91	0.79	0.91	0.77
	MVCGE-C	0.97	0.93	0.97	0.92
	MVCGE-SC	0.56	0.35	0.57	0.33
	MVCGE	1.00	1.00	1.00	1.00

6.2.5 Analysis of results and method comparison

According to Table 6.1, the performance of all multi-view clustering methods is better than that of SC -Best, which corresponds to the spectral clustering method applied to the best single view. In fact, the presence of multiple views brings additional information to the clustering method so that it can process the datasets better. The proposed method gives the best performance followed

by NESE, MVCSK, S-MVSC, CI-GMVC, MCLES and CNESE methods. With respect to the large MNIST dataset shown in Table 6.2, MVCGE shows similar results to CNESE for most cluster indicators. Moreover, the performance of our method is better than most competing methods for the same dataset, which shows that we are able to handle large datasets with this new approach and achieve good results. The results we obtained on the MNIST-1000 dataset (see Table 6.2) demonstrate the superiority of the proposed method. From Table 6.3, MVCGE achieves the best results for the synthetic datasets even when applied to the single view datasets.

6.2.6 Clustering visualization

In this section, we visualize the clustering obtained by the proposed MVCGE method on four datasets using the t-SNE technique [98]. In all subfigures of Figure 6.1, the spectral projection matrix \mathbf{P} and the nonnegative embedding matrix \mathbf{H} are shown for ORL, Tetra, Hepta and Chainlink. In these subfigures, each point corresponds to an image (ORL) or a 3D point (synthetic datasets). We emphasize that the color corresponds to the ground-truth classes.

For ORL we present five clusters. From Figures 6.1(a) and 6.1(b), it can be seen that Cluster 1 and Cluster 5 are not pure, as they each contain images associated with two different individuals, which explains the result obtained in Table 6.1. The clustering of the synthetic datasets of Tetra and Chainlink is shown in Figures 6.1(e), 6.1(f), 6.1(g) and 6.1(h). Some clustering errors are observed, which explain the results obtained in Table 6.3. Moreover, the visualization of the spectral representation and cluster index matrices (nonnegative embedding) associated with the Tetra dataset shows well-separated clusters in Figures 6.1(c) and 6.1(d). This confirms the perfect performance of 100 % in Table 6.3. Figure 6.2 shows the estimated two clusters obtained by MVCSK, NESE and MVCGE methods for the Chainlink dataset. According to this figure, the worst result is that of MVCSK and the best is that of our method. It is clear that the two clusters are well separated by using MVCGE and Figure 6.2(c) has few clustering errors.

6.3 Conclusion

A novel approach for multi-view clustering is proposed. Unlike existing methods, it simultaneously learns the unified similarity matrix, the uniform spectral projection matrix, the nonnegative embedding matrix (cluster index matrix) and the weight of each view. Thus, the final clustering result can be obtained directly from the nonnegative embedding matrix, which is a convolution of the consensus data representation over the graph. The proposed method combines the advantages

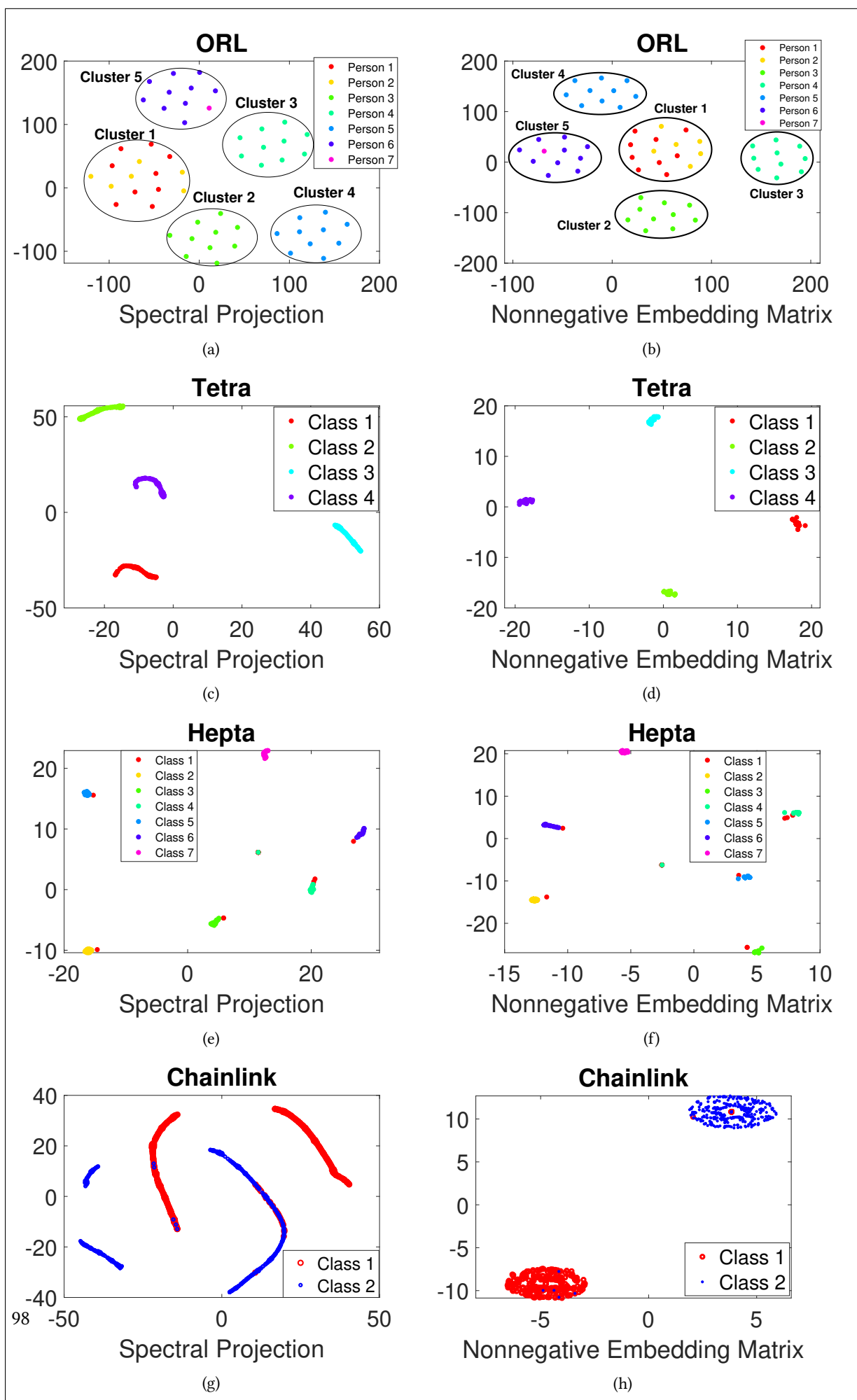


Figure 6.1: t-SNE of the spectral projection and nonnegative embedding matrices obtained by the proposed clustering method MVCGE for different datasets.

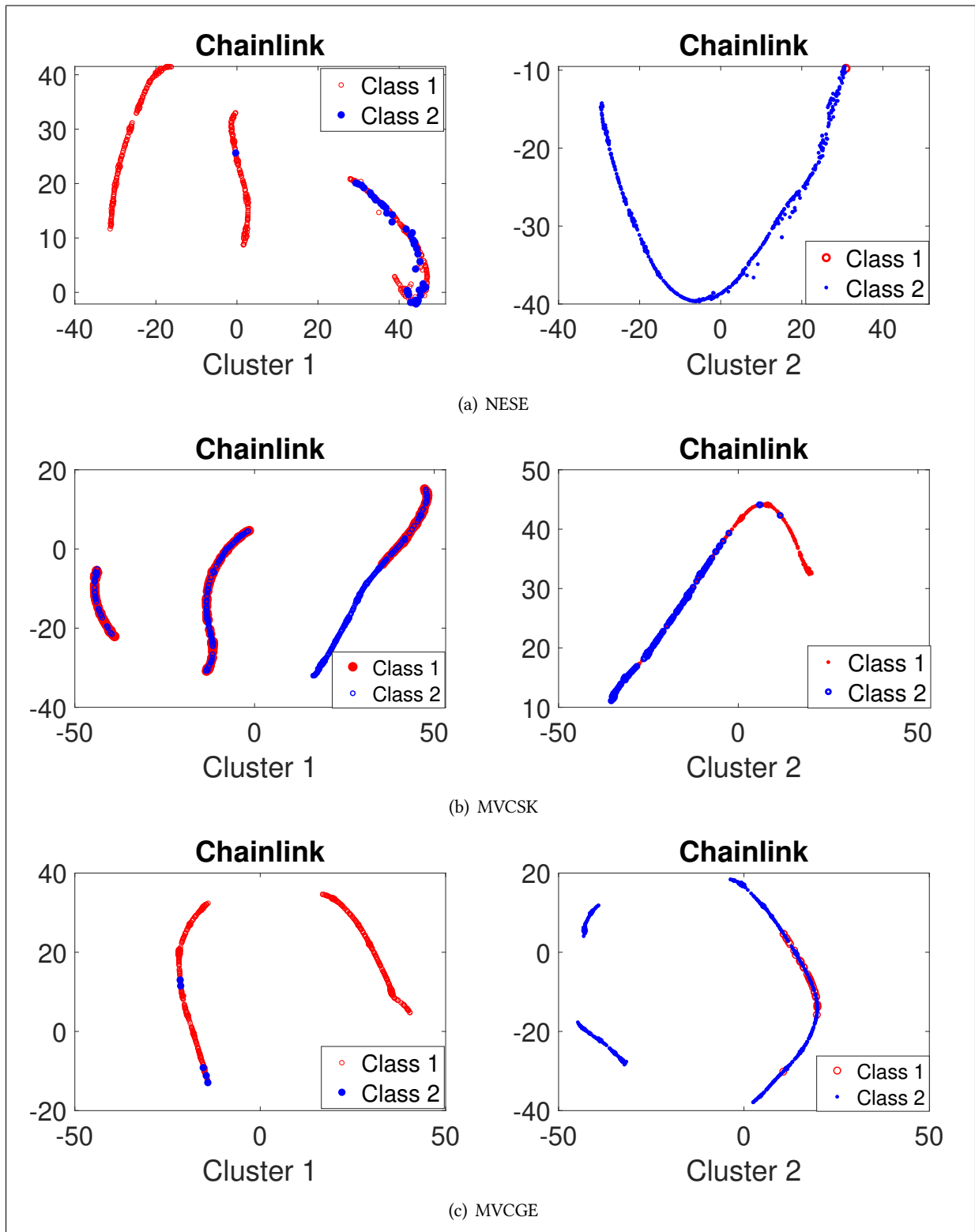


Figure 6.2: Visualization of the two clusters obtained by three different methods for the Chainlink dataset.

of graph-based approaches and matrix factorization-based methods. Experimental results on real and synthetic datasets have shown that MVCGE outperforms many state-of-the-art methods.

Multi-view Spectral Clustering via Constrained Nonnegative Embedding

This chapter presents a new multi-view spectral clustering via Constrained Nonnegative Embedding (CNESE) that can be considered as an improved version of the method "Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding" (NESE) [11]. *This method differs from the aforementioned methods (MKGNE and MVCGE), by the fact that each view is represented by a similarity matrix provided beforehand and used as input. Thus, the kernel matrices are not used in this method. Moreover, instead of artificially applying the orthogonality constraint to the nonnegative embedding matrix H , two additional terms are added to the objective function of our new model. These terms introduce two types of constraints on the nonnegative embedding matrix H : the first type of constraints is given by the smoothness of the cluster indices over the graphs; the second type of constraints is related to the orthogonality of the columns of the nonnegative embedding, which contributes to better cluster separation.* In addition, similar to the first two methods, our proposed method inherits the advantages of the NESE method, namely the simultaneous implementation of nonnegative embedding and spectral embedding matrices, which makes it possible to obtain clustering results directly without the need for a post-processing clustering method, such as k-means, or additional parameters.

The main contributions of this chapter are summarized as follows.

1. The proposed method retains the advantages of graph-based methods and matrix factorization-based methods.

2. Our method introduces a cluster indices smoothness term and an orthogonality constraint on the nonnegative embedding matrix. This constrained version of the method NESE can provide better clustering results than the original NESE.
3. It provides an efficient optimization scheme of the proposed criterion.
4. The proposed method provides the clustering results directly without any post processing procedures like k-means.

7.1 Proposed Approach

Inspired by NESE, in this chapter we develop a new method that can be considered as a constrained version of NESE and called "Multi-view spectral clustering via integrating a Constrained Nonnegative Embedding and Spectral Embedding" (CNESE). This method imposes a constraint on the nonnegative embedding matrix so that the clustering performance obtained can be better. It differs from the method NESE by adding a view-based label-like smoothness term and an orthogonality constraint on the nonnegative embedding matrix.

Given n data points with V feature vectors, the data matrix of each view can be denoted as: $\mathbf{X}^v = (\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v)$. Let $\mathbf{S}^v \in \mathbb{R}^{n \times n}$ be the similarity matrix of the view v . As in NESE, we want to estimate the spectral projection matrix $\mathbf{P}^v \in \mathbb{R}^{n \times C}$ of view v and the consistent nonnegative embedding matrix $\mathbf{H} \in \mathbb{R}^{n \times C}$, where C is the number of clusters. As we mentioned earlier, the objective function of NESE, is given by:

$$\min_{\mathbf{H}, \mathbf{P}^v} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2 \quad s.t. \quad \mathbf{H} \geq 0, \quad \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \quad (7.1)$$

In NESE, only the nonnegativity of the matrix \mathbf{H} is imposed. However, since the similarity graph of each view is available, we propose to use a set of additional constraints on the matrix \mathbf{H} . This is given by the cluster label smoothness over all views. Thus, by satisfying this smoothness condition two similar data points \mathbf{x}_i^v and \mathbf{x}_j^v in view v are forced to have similar cluster indices (i.e., if S_{ij}^v is large, then \mathbf{H}_{i*} should be close to \mathbf{H}_{j*}). For the view v , the term of cluster label smoothness to be minimized in the spectral analysis is given by the following equation:

$$\frac{1}{2} \sum_i \sum_j \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|_2^2 S_{ij}^v = Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H}), \quad (7.2)$$

where $\mathbf{L}^v \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the similarity matrix \mathbf{S}^v given by $\mathbf{D}^v - \mathbf{S}^v$, where \mathbf{D}^v is a diagonal matrix whose i -th diagonal element in the v -th view is given by: $D_{ii}^v = \sum_{j=1}^n \frac{S_{ij}^v + S_{ji}^v}{2}$.

In [99], the authors showed that imposing an orthogonality constraint on the soft label matrix can improve the results of semi-supervised classification. Inspired by [99], we enforce that the nonnegative embedding matrix \mathbf{H} has orthogonal columns. For simplicity, this orthogonality can be enforced by minimizing the following term:

$$\|\mathbf{H}^T \mathbf{H} - \mathbf{I}\|_2^2 = \text{Tr} \left((\mathbf{H}^T \mathbf{H} - \mathbf{I})^T (\mathbf{H}^T \mathbf{H} - \mathbf{I}) \right). \quad (7.3)$$

Finally, the objective function of our proposed learning model will be:

$$\begin{aligned} \min_{\mathbf{P}^v, \mathbf{H}} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2 + \lambda \sum_{v=1}^V \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})} + \alpha \text{Tr} \left((\mathbf{H}^T \mathbf{H} - \mathbf{I})^T (\mathbf{H}^T \mathbf{H} - \mathbf{I}) \right) \\ \text{s.t. } \mathbf{H} \geq 0, \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \end{aligned} \quad (7.4)$$

where λ is a regularization parameter, and α is a large positive value ensuring the orthogonality of the matrix \mathbf{H} .

Similar to methods that use view-based auto-weights [24, 95, 13, 100], we use two sets of view weights. These two sets correspond to the first and second terms in the objective function (7.4), respectively. The first set of weights is given by:

$$\delta_v = \frac{1}{2 * \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2} \quad v = 1, \dots, V. \quad (7.5)$$

The second set of weights associated with the smoothness terms of the views is given by:

$$w_v = \frac{1}{2 * \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} \quad v = 1, \dots, V. \quad (7.6)$$

Finally, by adopting these two sets of weights, it can be easily shown that the minimization problem presented in Eq. (7.4) is equivalent to minimizing the following objective function:

$$\begin{aligned} \min_{\mathbf{P}^v, \mathbf{H}} \sum_{v=1}^V \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2 + \lambda \sum_{v=1}^V w_v \text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H}) + \alpha \text{Tr} \left((\mathbf{H}^T \mathbf{H} - \mathbf{I})^T (\mathbf{H}^T \mathbf{H} - \mathbf{I}) \right) \\ \text{s.t. } \mathbf{H} \geq 0, \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \end{aligned} \quad (7.7)$$

Once we obtain the nonnegative embedding matrix \mathbf{H} , each instance is assigned to the cluster corresponding to the highest element in the row of that instance.

7.1.1 Optimization

In this section we describe an efficient optimization of the objective function in (7.7). We use an alternating minimization scheme to update the matrices \mathbf{H} and \mathbf{P}^v . It consists of fixing one matrix and updating the other. We repeat this process until convergence.

To initialize the matrix \mathbf{H} , we set both parameters λ and α to zero, so that we get the objective function of NESE. We use the efficient algorithm presented in [93] to compute an initial value for the matrix \mathbf{H} . We use the same schemes described in [93] to compute \mathbf{S}^v and to initialize \mathbf{P}^v .

Update \mathbf{P}^v : If we fix \mathbf{H} , w_v , and δ_v , the objective function of our method is equivalent to:

$$\min_{\mathbf{P}^v} \sum_{v=1}^V \delta_v \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2^2. \quad (7.8)$$

Since $\mathbf{P}^{vT}\mathbf{P}^v = \mathbf{I}$, this problem is the famous orthogonal Procrustes problem, and its solution can be obtained by using the singular value decomposition of $\mathbf{S}^{vT}\mathbf{H}$. Let $\mathbf{U}\Sigma\mathbf{V}^T = \text{SVD}(\mathbf{S}^{vT}\mathbf{H})$. The solution of equation (7.8) is given by:

$$\mathbf{P}^v = \mathbf{U}\mathbf{V}^T \quad \text{with} \quad \mathbf{U}\Sigma\mathbf{V}^T = \text{SVD}(\mathbf{S}^{vT}\mathbf{H}). \quad (7.9)$$

Update \mathbf{H} :

Fixing \mathbf{P}^v , w_v , and δ_v , we compute the derivative of the functional in (7.7) w.r.t. \mathbf{H} :

$$\frac{\partial f}{\partial \mathbf{H}} = 2 \sum_{v=1}^V \delta_v (\mathbf{H} - \mathbf{S}^v \mathbf{P}^v) + 2\lambda \sum_{v=1}^V w_v \mathbf{L}^v \mathbf{H} + 4\alpha \mathbf{H} (\mathbf{H}^T \mathbf{H} - \mathbf{I}).$$

Note that any real matrix \mathbf{A} can be written as the difference of two nonnegative matrices, i.e., $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$ where $\mathbf{A}^+ = \frac{1}{2}(|\mathbf{A}| + \mathbf{A})$ and $\mathbf{A}^- = \frac{1}{2}(|\mathbf{A}| - \mathbf{A})$. Let $\mathbf{M}^v = \mathbf{S}^v \mathbf{P}^v = \mathbf{M}^{v+} - \mathbf{M}^{v-}$, and $\mathbf{L}^v = \mathbf{L}^{v+} - \mathbf{L}^{v-}$.

After some algebraic manipulations, the gradient matrix can be written as: $\frac{\partial f}{\partial \mathbf{H}} = 2(\Delta^- - \Delta^+)$ where:

$$\Delta^- = \sum_{v=1}^V \delta_v \mathbf{H} + \sum_{v=1}^V \delta_v \mathbf{M}^{v-} + \lambda \sum_{v=1}^V w_v \mathbf{L}^{v+} \mathbf{H} + 2\alpha \mathbf{H} \mathbf{H}^T \mathbf{H}.$$

$$\Delta^+ = \sum_{v=1}^V \delta_v \mathbf{M}^{v+} + \lambda \sum_{v=1}^V w_v \mathbf{L}^{v-} \mathbf{H} + 2\alpha \mathbf{H}.$$

Using the gradient descent update rule, the nonnegative embedding matrix \mathbf{H} is updated as follows:

$$H_{ij} \leftarrow H_{ij} - \mu_{ij} \frac{\partial f}{\partial H_{ij}} = H_{ij} - \frac{1}{2\Delta_{ij}^-} H_{ij} * 2 * (\Delta_{ij}^- - \Delta_{ij}^+) = H_{ij} * \frac{\Delta_{ij}^+}{\Delta_{ij}^-}. \quad (7.10)$$

The learning parameter μ_{ij} is set to $\frac{1}{2\Delta_{ij}^-} H_{ij}$. Thus, the matrix \mathbf{H} can be updated with:

$$H_{ij} \leftarrow H_{ij} * \frac{\Delta_{ij}^+}{\Delta_{ij}^-} \quad i = 1, \dots, n; \quad j = 1, \dots, K. \quad (7.11)$$

Update w_v and δ_v :

After updating \mathbf{P}^v and \mathbf{H} , w_v and δ_v are updated using Equations (7.6) and (7.5), respectively.

The proposed CNESE method is summarized in **Algorithm 1**.

Algorithm 1 (CNESE)

Input:	Data samples $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$. The similarity matrix \mathbf{S}^v for each view. Parameters α and λ .
Output:	The consistent non negative embedding matrix \mathbf{H} . The spectral embedding matrix \mathbf{P}^v for each view.
Initialization:	The weights $w_v = \frac{1}{v}$ and $\delta_v = 1$. Initialize \mathbf{P}^v and \mathbf{H} as mentioned in section 3.1.
	Repeat
	Update \mathbf{P}^v , $v = 1, \dots, V$ using (7.9).
	Update \mathbf{H} using (7.11).
	Update w_v , $v = 1, \dots, V$ using (7.6).
	Update δ_v , $v = 1, \dots, V$ using (7.5).
	End

7.2 Performance Evaluation

7.2.1 Experimental Setup

To test the effectiveness of our method, eight image datasets were used. We compare our proposed method with several state-of-the-art methods, namely, the Co-training approach for multi-view Spectral Clustering (CotSC) [34], the Co-regularized approach for multi-view Spectral Clustering (CorSC) [35]. Also, some graph-based approaches are used such as: Multi-view Learning clustering with Adaptive Neighbors (MLAN) [44], Self-weighted Multi-view Clustering with multiple graphs (SwMC) [95], Affinity Aggregation for Spectral Clustering (AASC) [96], Graph Learning for Multi-View clustering (MVGL) [15], Parameter-free Auto-weighted Multiple Graph Learning (AMGL) [13], Multi-view clustering via Adaptively Weighted Procrustes (AWP) [20], Auto-weighted Multi-View Clustering via Kernelized graph learning (MVCSK) [24], and the method Multi-view spectral clustering via integrating Nonnegative Embedding and Spectral Embedding (NESE) [11]. The spectral clustering result of the best single view is also included and is denoted as "SC -best". Also, for some datasets, the average of the affinity matrices of all views is computed and then the Spectral Clustering algorithm is applied to this fused affinity matrix and denoted as "SC Fused".

In our method, the computation of the input matrices \mathbf{S}^v follows the same computational scheme as in [11]. \mathbf{H} is initialized as mentioned before.

Two parameters are used in our method: α and λ . The value of α and λ is chosen in the set $\{10, 10^{+2}, 10^{+3}, 10^{+4}, 10^{+5}, 10^{+6}, 10^{+7}, \text{ and } 10^{+8}\}$. It was shown that the best value of α is 10^{+6} (any other value larger than 10^{+6} does not improve the results).

7.2.2 Experimental results

Table 7.1 shows the results obtained by our method and several other methods on the COIL20, ORL, and Out-Scene datasets. The highest values are marked in bold. The number in parentheses shows the standard deviation of the indicator value calculated over several trials. These results show that the best clustering methods are MVSCK, NESE and CNESE (proposed). Therefore, we use these methods for comparison in the following. Table 7.2 shows a comparison between our methods and the state-of-the-art methods: NESE and MVCSK. The datasets used are: BBCSport, MSRCv1, MNIST-10000, and Extended Yale B Face. Table 7.3 shows a comparison between our method and some state-of-the-art methods on the Caltech101 dataset.

Table 7.1: Clustering performance on the COIL20, ORL, and Out-Scene datasets. The best performance for each indicator is in bold.

Dataset	Method	ACC	NMI	Purity	ARI
COIL20	SC-Best	0.73 (\pm 0.01)	0.82 (\pm 0.01)	0.75 (\pm 0.01)	0.68 (\pm 0.02)
	AWP	0.68 (\pm 0.00)	0.87 (\pm 0.00)	0.75 (\pm 0.00)	0.71 (\pm 0.00)
	MLAN	0.84 (\pm 0.00)	0.92 (\pm 0.00)	0.88 (\pm 0.00)	0.81 (\pm 0.00)
	SwMC	0.86 (\pm 0.00)	0.94 (\pm 0.00)	0.90 (\pm 0.00)	0.84 (\pm 0.00)
	AMGL	0.80 (\pm 0.04)	0.91 (\pm 0.02)	0.85 (\pm 0.03)	0.74 (\pm 0.07)
	AASC	0.79 (\pm 0.00)	0.89 (\pm 0.00)	0.83 (\pm 0.00)	0.76 (\pm 0.00)
	MVGL	0.78 (\pm 0.00)	0.88 (\pm 0.00)	0.81 (\pm 0.00)	0.75 (\pm 0.00)
	CorSC	0.68 (\pm 0.04)	0.78 (\pm 0.02)	0.70 (\pm 0.03)	0.62 (\pm 0.03)
	CotSC	0.70 (\pm 0.03)	0.80 (\pm 0.02)	0.72 (\pm 0.03)	0.65 (\pm 0.03)
	MVCSK	0.65 (\pm 0.04)	0.80 (\pm 0.02)	0.70 (\pm 0.03)	0.61 (\pm 0.05)
	NESE	0.77 (\pm 0.00)	0.88 (\pm 0.00)	0.82 (\pm 0.00)	0.69 (\pm 0.00)
	CNESE	0.82 (\pm 0.00)	0.88 (\pm 0.00)	0.82 (\pm 0.00)	0.78 (\pm 0.00)
ORL	SC-Best	0.66 (\pm 0.02)	0.76 (\pm 0.02)	0.71 (\pm 0.02)	0.67 (\pm 0.01)
	AWP	0.80 (\pm 0.00)	0.91 (\pm 0.00)	0.83 (\pm 0.00)	0.76 (\pm 0.00)
	MLAN	0.78 (\pm 0.00)	0.88 (\pm 0.00)	0.82 (\pm 0.00)	0.67 (\pm 0.00)
	SwMC	0.77 (\pm 0.00)	0.90 (\pm 0.00)	0.83 (\pm 0.00)	0.62 (\pm 0.00)
	AMGL	0.75 (\pm 0.02)	0.90 (\pm 0.02)	0.82 (\pm 0.02)	0.63 (\pm 0.09)
	AASC	0.82 (\pm 0.02)	0.91 (\pm 0.01)	0.85 (\pm 0.01)	0.76 (\pm 0.02)
	MVGL	0.75 (\pm 0.00)	0.88 (\pm 0.00)	0.80 (\pm 0.00)	0.55 (\pm 0.00)
	CorSC	0.77 (\pm 0.03)	0.90 (\pm 0.01)	0.82 (\pm 0.03)	0.72 (\pm 0.04)
	CotSC	0.75 (\pm 0.04)	0.87 (\pm 0.01)	0.78 (\pm 0.03)	0.67 (\pm 0.03)
	MVCSK	0.85 (\pm 0.02)	0.94 (\pm 0.01)	0.88 (\pm 0.02)	0.81 (\pm 0.02)
	NESE	0.82 (\pm 0.00)	0.91 (\pm 0.00)	0.85 (\pm 0.00)	0.75 (\pm 0.00)
	CNESE	0.87 (\pm 0.00)	0.95 (\pm 0.00)	0.89 (\pm 0.00)	0.84 (\pm 0.00)
Out-Scene	SC-best	0.47 (\pm 0.01)	0.39 (\pm 0.01)	0.57 (\pm 0.01)	0.34 (\pm 0.01)
	AWP	0.65 (\pm 0.00)	0.51 (\pm 0.00)	0.65 (\pm 0.00)	0.42 (\pm 0.00)
	MLAN	0.55 (\pm 0.02)	0.47 (\pm 0.01)	0.55 (\pm 0.02)	0.33 (\pm 0.03)
	SwMC	0.50 (\pm 0.00)	0.47 (\pm 0.00)	0.50 (\pm 0.00)	0.38 (\pm 0.00)
	AMGL	0.51 (\pm 0.05)	0.45 (\pm 0.03)	0.52 (\pm 0.04)	0.34 (\pm 0.05)
	AASC	0.60 (\pm 0.00)	0.48 (\pm 0.00)	0.60 (\pm 0.00)	0.35 (\pm 0.00)
	MVGL	0.42 (\pm 0.00)	0.31 (\pm 0.00)	0.43 (\pm 0.00)	0.16 (\pm 0.00)
	CorSC	0.51 (\pm 0.04)	0.39 (\pm 0.03)	0.52 (\pm 0.03)	0.31 (\pm 0.02)
	CotSC	0.38 (\pm 0.02)	0.22 (\pm 0.01)	0.39 (\pm 0.02)	0.16 (\pm 0.01)
	MVCSK	0.65 (\pm 0.01)	0.52 (\pm 0.00)	0.65 (\pm 0.01)	0.42 (\pm 0.00)
	NESE	0.63 (\pm 0.00)	0.53 (\pm 0.00)	0.66 (\pm 0.00)	0.46 (\pm 0.00)
	CNESE	0.66 (\pm 0.00)	0.55 (\pm 0.00)	0.67 (\pm 0.00)	0.47 (\pm 0.00)

Table 7.2: Clustering performance on the MSRCv1, BBCSport, Extended-Yale, and MNIST-10000 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
MSRCv1	SC Fused	0.77 (\pm 0.00)	0.70 (\pm 0.00)	0.79 (\pm 0.00)	0.61 (\pm 0.00)
	MVCSK	0.70 (\pm 0.02)	0.59 (\pm 0.03)	0.70 (\pm 0.02)	0.50 (\pm 0.04)
	NESE	0.77 (\pm 0.00)	0.72 (\pm 0.00)	0.80 (\pm 0.00)	0.64 (\pm 0.00)
	CNESE	0.86 (\pm 0.00)	0.76 (\pm 0.00)	0.86 (\pm 0.00)	0.72 (\pm 0.00)
BBCSport	SC Fused	0.72 (\pm 0.06)	0.60 (\pm 0.04)	0.72 (\pm 0.04)	0.48 (\pm 0.00)
	MVCSK	0.90 (\pm 0.07)	0.82 (\pm 0.02)	0.90 (\pm 0.02)	0.85 (\pm 0.07)
	NESE	0.72 (\pm 0.00)	0.69 (\pm 0.00)	0.75 (\pm 0.00)	0.60 (\pm 0.00)
	CNESE	0.72 (\pm 0.00)	0.68 (\pm 0.00)	0.76 (\pm 0.00)	0.60 (\pm 0.00)
Ext-Yale	SC Fused	0.36 (\pm 0.01)	0.49 (\pm 0.02)	0.40 (\pm 0.01)	0.14 (\pm 0.02)
	MVCSK	0.33 (\pm 0.00)	0.42 (\pm 0.00)	0.34 (\pm 0.00)	0.18 (\pm 0.00)
	NESE	0.43 (\pm 0.00)	0.58 (\pm 0.00)	0.47 (\pm 0.00)	0.25 (\pm 0.00)
	CNESE	0.60 (\pm 0.00)	0.75 (\pm 0.00)	0.60 (\pm 0.00)	0.51 (\pm 0.00)
MNIST -10000	SC Fused	0.20 (\pm 0.00)	0.13 (\pm 0.00)	0.20 (\pm 0.00)	0.05 (\pm 0.00)
	MVCSK	0.49 (\pm 0.00)	0.41 (\pm 0.00)	0.50 (\pm 0.00)	0.29 (\pm 0.00)
	NESE	0.81 (\pm 0.00)	0.83 (\pm 0.00)	0.85 (\pm 0.00)	0.76 (\pm 0.00)
	CNESE	0.81 (\pm 0.00)	0.83 (\pm 0.00)	0.86 (\pm 0.00)	0.78 (\pm 0.00)

Table 7.3: Clustering performance on the Caltech101 dataset. The best performance for each indicator is bolded. The "-" symbol indicates that the ARI indicator was not provided by the corresponding published paper.

Dataset	Method	ACC	NMI	Purity	ARI
Caltech101	SC Best	0.10 (\pm 0.00)	0.32 (\pm 0.01)	0.22 (\pm 0.00)	-
	AMGL	0.22 (\pm 0.00)	0.38 (\pm 0.01)	0.40 (\pm 0.00)	-
	MLAN	0.22 (\pm 0.00)	0.31 (\pm 0.01)	0.37 (\pm 0.00)	-
	MVCSK	0.17 (\pm 0.00)	0.20 (\pm 0.01)	0.18 (\pm 0.00)	0.03 (\pm 0.01)
	NESE	0.30 (\pm 0.00)	0.47 (\pm 0.00)	0.46 (\pm 0.00)	0.19 (\pm 0.00)
	CNESE	0.32 (\pm 0.00)	0.49 (\pm 0.00)	0.48 (\pm 0.00)	0.21 (\pm 0.00)

7.2.3 Parameter sensitivity and ablation study

As mentioned earlier, two parameters are used in our approach: α , and λ . In theory, a large value of α is preferred. In our experiments, the value of α and the value of λ vary within the set $\{10, 10^{+2}, 10^{+3}, 10^{+4}, 10^{+5}, 10^{+6}, 10^{+7}, \text{ and } 10^{+8}\}$. Figure 7.1 plots the ACC and NMI indicators as a function of α and λ . The plots shown correspond to four datasets: COIL20, ORL, MSRCv1, and BBCSport. From this figure, it can be seen that for most datasets, any value of α less than 10^{+6} leads to low clustering performance; moreover, any value greater than 10^{+6} leads to results almost equal to those obtained considering α equal to 10^{+6} , for which the clustering performance is high. In this setting, we did not observe any performance improvement by further increasing α . Therefore, a value of α equal to 10^{+6} will be the best choice.

If we assume that α is fixed at 10^{+6} , according to this figure, the best values of ACC and NMI are obtained for a value of λ equal to 10^{+3} , 10^{+7} , 10^{+5} and 10^{+6} , respectively, for the COIL20, ORL, MSRCv1 and BBCSport datasets. In this setting (α is fixed at 10^{+6}), it can be seen that the clustering performance is relatively stable when λ varies.

We also perform an ablation study using the proposed CNESE method. Specifically, we define

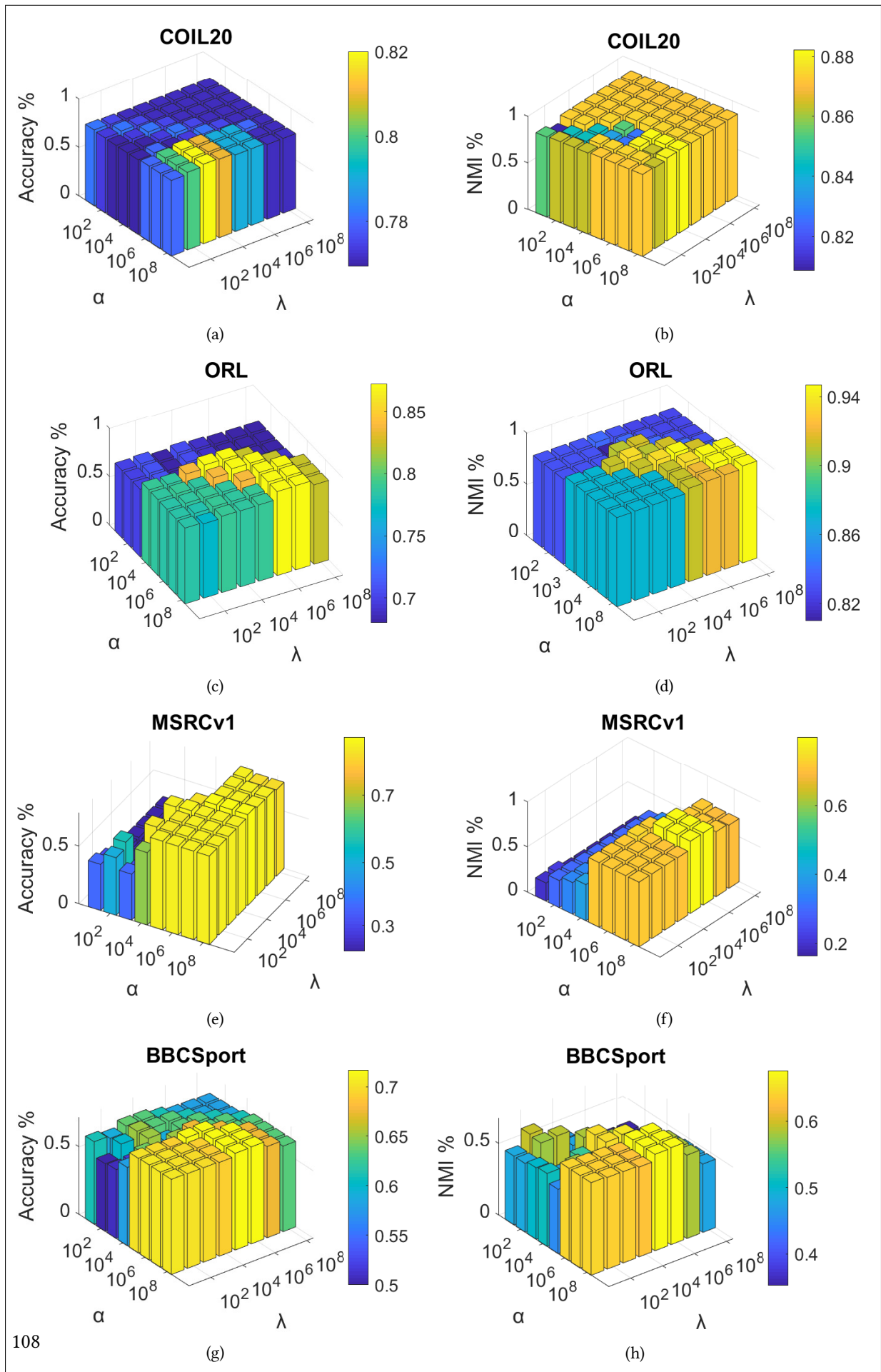


Figure 7.1: Clustering performance ACC (%) (left column) and NMI (%) (right column) as a function of α and λ on the COIL20, ORL, MSRCv1, and BBCSport datasets.

two different variants of CNESE: (1) No smoothness term in the global objective function (7.7) (i.e., λ is set to zero), and we call the resulting method CNESE-O, which means that only the orthogonality constraint is used in CNESE. (2) No orthogonality constraint (α is set to zero), and we call the resulting method CNESE-R. The results obtained with CNESE-O, CNESE-R, and CNESE are summarized in Table 7.4. We used seven different datasets. As shown in this table, both the smoothness term and the orthogonality constraint contributed to the good clustering performance of the proposed method.

Table 7.4: Ablation study with different conditions. The best performance for each indicator is in bold.

Dataset	Variant	ACC	NMI	Purity	ARI
ORL	NESE	0.82	0.91	0.85	0.75
	CNESE-O	0.82	0.91	0.85	0.76
	CNESE-R	0.83	0.91	0.86	0.76
	CNESE	0.87	0.95	0.89	0.84
MSRCv1	NESE	0.77	0.72	0.80	0.64
	CNESE-O	0.77	0.72	0.80	0.64
	CNESE-R	0.77	0.72	0.80	0.64
	CNESE	0.86	0.76	0.86	0.72
Out-scene	NESE	0.63	0.53	0.66	0.46
	CNESE-O	0.62	0.51	0.65	0.45
	CNESE-R	0.63	0.53	0.66	0.47
	CNESE	0.66	0.55	0.67	0.47
BBCSport	NESE	0.72	0.69	0.75	0.60
	CNESE-O	0.72	0.69	0.75	0.60
	CNESE-R	0.72	0.69	0.75	0.60
	CNESE	0.72	0.68	0.76	0.60
Exended-Yale	NESE	0.43	0.58	0.47	0.25
	CNESE-O	0.43	0.57	0.46	0.24
	CNESE-R	0.45	0.59	0.47	0.25
	CNESE	0.60	0.75	0.60	0.51
MNIST-10000	NESE	0.81	0.83	0.85	0.76
	CNESE-O	0.81	0.83	0.85	0.76
	CNESE-R	0.81	0.83	0.85	0.76
	CNESE	0.81	0.83	0.86	0.78
Caltech101	NESE	0.30	0.47	0.46	0.19
	CNESE-O	0.30	0.47	0.46	0.19
	CNESE-R	0.30	0.47	0.46	0.19
	CNESE	0.32	0.49	0.48	0.21

7.2.4 Analysis of results and method comparison

The previously mentioned tables and figures summarize the experimental results of our method. According to Table 7.1, the performance of the spectral clustering algorithm applied to a single view, referred to as SC Best, is always lower than the performance of other algorithms applied to multi-view datasets. This is due to the fact that using multiple views provides additional information that can improve the results. From Tables 7.1, 7.2, and 7.3, we can infer that the results obtained by our method are superior to those obtained by other methods for most datasets. In particular, the performance indicators, namely ACC, NMI, purity and ARI, obtained by our

proposed method are greater than those obtained by the NESE method for all datasets. For the COIL20 dataset, although the indicators obtained by CNESE are lower than those obtained by SwMC, they are greater than those obtained by NESE, which shows that our method, which is the constrained version of NESE, has improved the result. Although the performance of CNESE for the BBCSport dataset is worse than that of MVCSK for all clustering indicators, it is still equal or close to the performance obtained by NESE for the four indicators.

Regarding the Extended Yale dataset, the high results of CNESE can be interpreted as follows: This dataset consists of face images with large illumination variations. When we compare the results of NESE, MVCSK and SC Fused with those of CNESE, we can find that our method achieves better clustering results than other multi-view clustering methods. Therefore, the competing methods, SC Fused, MVCSK and NESE could not take advantage of the two views (two types of descriptors). On the other hand, the proposed CNESE enforces an automatically weighted view-based smoothness on the cluster index matrix as well as an orthogonality constraint, which justifies its high performance. This is also confirmed in the ablation study presented in Table 7.4 (Extended Yale dataset). For SC Fused, the use of a unified graph created by fusing the individual graphs of the Extended Yale dataset resulted in low performance. For example, the ACC indicator changes from 36% to 60% when switching from SC Fused to CNESE.

The proposed method was tested on two large datasets MNIST-10000 and Caltech101. For the large MNIST-10000 dataset, Table 7.2 shows that the performance obtained for CNESE is very close to or even exceeds that of NESE, but is still significantly better than the MVCSK and SC methods. Moreover, Table 7.3 shows that our method gave better performance compared to other approaches for the large dataset Caltech101 .

The obtained results show that the smoothness of the cluster indices over the graphs and the orthogonality of the cluster index matrix contributed to better cluster separation.

7.2.5 Convergence study

In this section, the convergence of CNESE is studied. The number of iterations is fixed at 40. For each iteration, the value of the objective function is computed. Our objective function is given by:

$$\sum_{v=1}^V \|\mathbf{s}^v - \mathbf{H}\mathbf{P}^{vT}\|_2 + \lambda \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H})} + \alpha \text{Tr}((\mathbf{H}^T \mathbf{H} - \mathbf{I})^T (\mathbf{H}^T \mathbf{H} - \mathbf{I})).$$

Figure 7.2 shows the convergence of our method on the MSRCv1 dataset. The "x" axis in this figure represents the number of iterations and the "y" axis represents the value of the objective function computed at each iteration. Figure 7.2(a) represents the convergence of CNESE on the

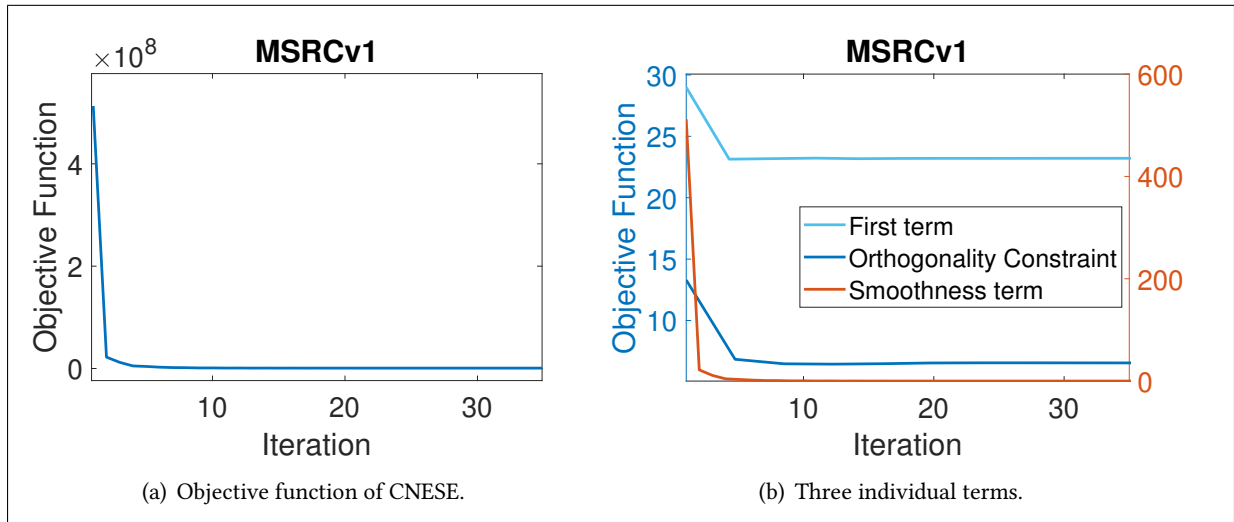


Figure 7.2: Convergence of CNESE on the MSRCv1 dataset.

MSRCv1 dataset, and Figure 7.2(b) shows the convergence of the individual terms of the objective function of CNESE. As can be seen from Figure 7.2, our method converges quickly. The solution was obtained in only 10 iterations.

7.2.6 Computational complexity analysis and time cost

In this section, we study the computational complexity of the proposed method. Our algorithm is based on four main steps: Updating \mathbf{H} , \mathbf{P} , w_v and δ_v (see **Algorithm 1**). Looking at the four steps of **Algorithm 1** from a computational point of view, it is easy to see that step 1 is the most expensive one. In fact, Step 2, Step 3 and Step 4 are based on simple matrix multiplications and additions. Therefore, we can ignore their contribution to the computational cost of **Algorithm 1** in favor of the first step.

To estimate the matrix \mathbf{P} (Step 1), we should compute the SVD of the matrix $(\mathbf{S}^{vT} \mathbf{H})$. The computational cost of this is $\mathcal{O}(nC^2)$, where C is the total number of clusters. Since $C \ll n$, the computational complexity of our algorithm is lower than that of most graph-based algorithms which is equal to $\mathcal{O}(n^3)$. It should be noted that the computational complexities of the main competing methods MVSCK and NESE are $\mathcal{O}(n^3)$ and $\mathcal{O}(nC^2)$ respectively, while that of many competing graph-based methods is $\mathcal{O}(n^3)$.

In Table 7.5, we compare the time cost of MVSCK, NESE, and CNESE. For these three methods, we set the number of iterations to 60.

From Table 7.5, we can observe that the time cost of our method is slightly higher than that of NESE for all datasets. In fact, the CNESE method computes the cluster index matrix differently

Table 7.5: The time cost (seconds) of MVCSK, NESE, and CNESE.

Dataset	ORL	COIL20	MSRCv1	BBCSport
MVCSK	80.33	2667.30	8.34	168.29
NESE	3.37	24.26	0.79	49.41
CNESE (ours)	4.34	26.98	1.07	52.60

in each iteration. Moreover, it has some additional processes related to the estimation of the additional automatic weights. In [11], it was shown that the NESE method is one of the fastest clustering algorithms. In many datasets, it was even ranked as the second or third fastest algorithm among the baselines CotSC, CorSC, MLAN, SwMC, AASC, MVGL, AMGL and AWP.

On the other hand, it can be seen from Table 7.5 that the time cost of our method is lower than that of MVCSK for all datasets. This result can be explained by the fact that MVCSK is a graph-based method with computational complexity equal to $\mathcal{O}(n^3)$. Moreover, it has additional cost associated with the k-means clustering step.

7.3 Conclusion

In this chapter, we presented a constrained version of a recent graph-based clustering approach called Nonnegative Embedding and Spectral Embedding. To improve the multi-view clustering performance of this approach, we introduced a new criterion that uses two constraints on the nonnegative embedding matrix (cluster index matrix). The first constraint is given by the smoothness of the cluster indices over the graphs. The second constraint is given by the orthogonality of the cluster index matrix.

We have studied the effect of these two constraints on the final clustering performance. Extensive experimental results show the effectiveness of the proposed method. Experiments on real image datasets have shown that adopting the proposed constraints in the model improves the clustering results. The proposed approach can be applied to different types and sizes of datasets.

One-step Multi-view Spectral Clustering with Cluster Label Correlation Graph

In this chapter, we present a novel approach to one-step graph-based multi-view clustering. This method is called Multi-view Spectral Clustering with a Self-Taught Robust Graph Learning (MCSRGL). *In contrast to the three aforementioned multi-view clustering methods, our proposed method introduces a novel innovation, inspired by semi-supervised learning. This key innovation consists of creating an additional graph by using the cluster label correlation to the graphs associated with the data space. Second, similar to the CNESE method, a smoothing constraint is exploited to constrain the cluster-label matrix and make it more consistent with the original data graphs as well as with the label graph. Moreover, this method considers the same input data graphs adopted by CNESE to represent each view. Thus, the kernel representation is not adopted in our model.* Experimental results on several public datasets show the efficiency of the proposed approach. All cluster evaluation metrics show significant improvement by applying our method to different types and sizes of datasets. The average improvement (across all datasets) is the difference between the indicator obtained by our approach and the indicator obtained by the most competitive method. The average improvement is approximately 4 %, 2 %, 3 %, and 2 % for the Accuracy indicator, the Normalized Mutual Information indicator, the Purity indicator, and the Adjusted Rand index, respectively. Therefore, our new method brings two main improvements to the NESE method presented in [11]. The first improvement is provided by a smoothness constraint on the cluster label matrix, and the second

improvement is achieved by an additional graph built upon the estimated soft cluster labels. The smoothness is imposed by the multiple-view graphs as well as by the adaptive label graph.

The contributions of this chapter are as follows.

1. Unlike other multi-view clustering approaches, we proposed a new approach that can provide an additional graph-based on the soft cluster labels.
2. This method introduces cluster label smoothness which improves the performance.
3. This approach provides the final clustering assignment without a post-processing step such as k-means or spectral rotation.
4. An efficient optimization scheme is introduced to solve the proposed criterion.
5. Several experiments are conducted on different types and sizes of datasets and show the superiority of the proposed approach.

8.1 Proposed Approach

This chapter develops a novel approach inspired by NESE. This method is called Multi-view Spectral Clustering with a self-taught Robust Graph Learning (MCSRGL). First, this method introduces a label-like smoothness constraint on the nonnegative embedding matrix (cluster label matrix). Second, an additional graph built from the space of labels is added in the form of an additional view to add richness to the data and improve the results. Given a data matrix $\mathbf{X}^v = (\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v)$, n is the number of samples, and V is the total number of views. For a given view v , $\mathbf{S}^v \in \mathbb{R}^{n \times n}$ is the similarity matrix, $\mathbf{P}^v \in \mathbb{R}^{n \times C}$ is the spectral representation matrix associated with the similarity matrix \mathbf{S}^v , and $\mathbf{H} \in \mathbb{R}^{n \times C}$ is the consensus cluster label matrix used for clustering. Note that C is the number of clusters. As we mentioned earlier, our new method is an improvement of the method in [11], so the first term of our method is the same as that of the method NESE and is given by:

$$\min_{\mathbf{H}, \mathbf{P}^v} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2 \quad s.t. \quad \mathbf{H} \geq 0, \quad \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \quad (8.1)$$

Unlike NESE, which imposes only a non-negativity constraint on the matrix \mathbf{H} , our method adds a cluster label smoothness term over all views. This smoothness constraint can be interpreted as follows: Given two data samples \mathbf{x}_i^v and \mathbf{x}_j^v in view v , if \mathbf{x}_i^v and \mathbf{x}_j^v are similar (i.e., the value of the element S_{ij}^v is large), they are forced to have similar cluster labels (i.e., \mathbf{h}_i should be very close

to \mathbf{h}_j). Here, \mathbf{h}_i (a C-vector) denotes the i -th row of \mathbf{H} and is the cluster label vector of the i -th sample. Moreover, the second term of our method will be given by:

$$\frac{1}{2} \sum_i \sum_j \|\mathbf{h}_i - \mathbf{h}_j\|_2^2 S_{ij}^v = Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H}), \quad (8.2)$$

where $\mathbf{L}^v = \mathbf{D}^v - \mathbf{S}^v \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the associated similarity matrix \mathbf{S}^v , \mathbf{D}^v is the diagonal matrix of view v with i -th element given by: $D_{ii}^v = \sum_{j=1}^n \frac{S_{ij}^v + S_{ji}^v}{2}$.

The objective function of our proposed model will be:

$$\min_{\mathbf{P}^v, \mathbf{H}} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2 + \lambda \sum_{v=1}^V \sqrt{Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H})} \quad s.t. \quad \mathbf{H} \geq 0, \quad \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}, \quad (8.3)$$

where λ is a regularization parameter.

To control the importance of each view in global optimization, we use the automatic weighting of views described in the literature [24, 44, 13, 43, 23]. In our criterion, the weights of the data views as well as the additional view (based on the estimated labels) are adaptively set and updated, taking into account the contribution of each view to the loss terms. Thus, they can reduce the impact of noisy views.

Making the weight inversely proportional to the square root of the loss was first introduced in [13]. Since then, this iterative trick has been used in many works aimed at minimizing an objective function consisting of an additive aggregation of multiple losses or terms.

As we have already mentioned, this idea of using an automatically weighted strategy to minimize an additive function of multiple losses was inspired by previous works to avoid the use of extra predefined parameters and reduce the complexity of the proposed method, making the optimization scheme simpler and more efficient. The square root of the Frobenius norm and the trace term in (8.3) allows us to derive these automatic weights by taking the derivatives of the objective function. For this purpose, two sets of weights are added. The first set of weights is assigned to the first term of our objective function, and the second set of weights is assigned to the smoothness term. These weights are automatically updated during the iterative process of the proposed method.

The first set of weights is given by the following equation:

$$\delta_v = \frac{1}{2 * \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2} \quad v = 1, \dots, V. \quad (8.4)$$

The second set of weights is given by:

$$w_v = \frac{1}{2 * \sqrt{Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} \quad v = 1, \dots, V. \quad (8.5)$$

Thus, the objective function of MCSRGL is given by:

$$\begin{aligned} \min_{\mathbf{P}^v, \mathbf{H}} \sum_{v=1}^V \delta_v \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2^2 + \lambda \sum_{v=1}^V w_v \text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H}) \\ \text{s.t. } \mathbf{H} \geq 0, \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \end{aligned} \quad (8.6)$$

8.1.1 Incorporating cluster label space

Most multi-view clustering algorithms extract information from the data space and ignore the cluster labels. The data space, also known as a feature descriptor, extracts data information. Relying only on the graphs derived from the features, without using the cluster memberships (in the form of predictions), may not be the best option for a clustering task. Most current graph-based algorithms produce graphs from the original feature space, which may be sensitive to noise or outliers. We introduce a new similarity metric based on the cluster label space. We consider the cluster label space as a new way of looking at data that can be characterized by a new similarity metric. Indeed, the labeling information can be used to create an additional graph that can be integrated into the multi-view clustering criterion. In our proposed approach, this additional graph is constructed using the correlation of the predicted labels. Therefore, if two samples that have low similarity in the data space may have high similarity in the cluster label space. The cluster label space can affect the similarity between data points. The basic problem is to figure out how to combine the cluster label space and the data space.

However, the methods in [101, 102, 103] show that the label space contains hidden information that can improve the classification performance. Therefore, using the label space can provide an additional similarity matrix representing the data. Thus, this extra graph can be used as an extra view in the model (8.3).

The principle of this new method is to create a label graph and assign it to an additional view. We will have a number of views equal to $V + 1$. Let \mathbf{S}^{V+1} denotes this additional graph. The nodes of this graph are the data points and the weight of each edge is the Pearson correlation coefficient, as shown in Eq. (8.7). Given two data points \mathbf{x}_i and \mathbf{x}_j with corresponding label vectors \mathbf{h}_i and \mathbf{h}_j , the similarity between these data points is given by Eq. (8.7).

$$S^{V+1}(i, j) = \text{correlation}(\mathbf{h}_i, \mathbf{h}_j) = \frac{\sum_{k=1}^C (h_{ik} - m_i)(h_{jk} - m_j)}{\sqrt{\sum_{k=1}^C (h_{ik} - m_i)^2} \sqrt{\sum_{k=1}^C (h_{jk} - m_j)^2}}, \quad (8.7)$$

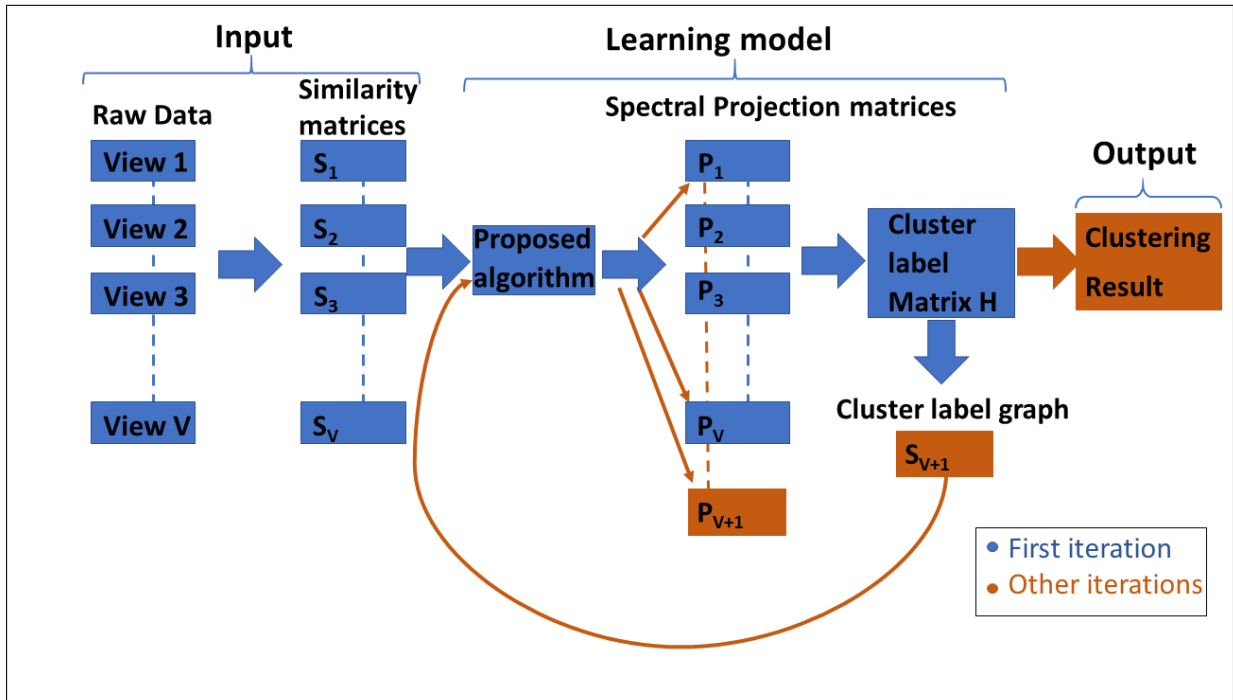


Figure 8.1: Illustration of the MCSRGL method.

where m_i and m_j are the mean values of row vectors \mathbf{h}_i and \mathbf{h}_j , respectively. The value of this correlation coefficient varies between -1 and $+1$. If the two vectors \mathbf{h}_i and \mathbf{h}_j are equal, this value is $+1$. A value of $+1$ means a positive perfect correlation and a value of -1 means a negative perfect correlation. If the two vectors \mathbf{h}_i and \mathbf{h}_j are completely uncorrelated, the correlation coefficient is zero.

Once the n^2 correlation coefficients are estimated, we eliminate the negative values. The obtained matrix is too dense to reveal the cluster structure, so we further build a p -nearest neighbor graph matrix. Specifically, we only retain the first p largest similarities for each sample, and set the others to zero. This graph forms an additional view $V + 1$ that is integrated with the graphs of the other views.

The illustration of the proposed method is shown in Figure 8.1. From this figure, it can be seen that after estimating the matrix \mathbf{H} , the label graph is computed and an additional view related to the label space is added during the iterations of the algorithm. Then the final objective function of the learning model is given below.

$$\min_{\mathbf{P}^v, \mathbf{H}} \sum_{v=1}^{V+1} \delta_v \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2^2 + \lambda \sum_{v=1}^{V+1} w_v \text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H})$$

$$s.t. \mathbf{H} \geq 0, \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \quad (8.8)$$

In addition, the equations of the weights will be given again by Eqs. (8.4) and (8.5) where

$v = 1, \dots, V + 1$.

8.1.2 Optimization

In this section, we present the procedure for optimizing the objective function in (8.8). We use an effective algorithm based on an alternating minimization scheme to solve the final objective function and update the matrices \mathbf{H} and \mathbf{P}^v .

To initialize the matrices \mathbf{S}^v and their corresponding spectral representations \mathbf{P}^v , the same method as in [93] is used. Moreover, as mentioned earlier, to initialize the matrix \mathbf{S}^{V+1} , the \mathbf{H} step (see Eq. (8.9)) of our algorithm is used with a number of views equal to V , then \mathbf{S}^{V+1} is computed using the correlation coefficients of the rows of \mathbf{H} as explained in the previous section. The initial spectral representation, \mathbf{P}^{V+1} , is set to the C eigenvectors of \mathbf{S}^{V+1} associated with the C largest eigenvalues.

Then, the algorithm iteratively performs the following two steps alternately.

Update \mathbf{H} : Fixing \mathbf{P}^v , w_v and δ_v , we compute the derivative of the functional in (8.8) with respect to \mathbf{H} :

$$\frac{\partial f}{\partial \mathbf{H}} = \sum_{v=1}^{V+1} 2 \delta_v (\mathbf{H} - \mathbf{S}^v \mathbf{P}^v) + 2 \lambda \sum_{v=1}^{V+1} w_v \mathbf{L}^v \mathbf{H}.$$

The optimal solution \mathbf{H} is obtained by vanishing this derivative. Consequently, \mathbf{H} is given by:

$$\mathbf{H} = \left(\sum_{v=1}^{V+1} (\delta_v \mathbf{I} + \lambda w_v \mathbf{L}^v) \right)^{-1} \left(\sum_{v=1}^{V+1} \delta_v \mathbf{S}^v \mathbf{P}^v \right). \quad (8.9)$$

Thus, to obtain the matrix \mathbf{H} , we apply the element-wise ReLU (Rectified Linear Unit) operator to the elements of the matrix \mathbf{H} obtained by Equation (8.9).

Update \mathbf{P}^v : Fixing \mathbf{H} , w_v and δ_v , the objective function of our method is equivalent to:

$$\min_{\mathbf{P}^v} \sum_{v=1}^{V+1} \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2. \quad (8.10)$$

Given that $\mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}$, the above problem is the famous orthogonal Procrustes problem. The solution of this problem can be obtained using the singular value decomposition of $\mathbf{S}^{vT} \mathbf{H}$. Let $\mathbf{Y} \mathbf{\Sigma} \mathbf{T}^T = \text{SVD}(\mathbf{S}^{vT} \mathbf{H})$. Thus, the solution of the equation (8.10) is given by:

$$\mathbf{P}^v = \mathbf{Y} \mathbf{T}^T \quad \text{with} \quad \mathbf{Y} \mathbf{\Sigma} \mathbf{T}^T = \text{SVD}(\mathbf{S}^{vT} \mathbf{H}). \quad (8.11)$$

Update w_v and δ_v :

After updating the two matrices \mathbf{P}^v and \mathbf{H} , the weights δ_v and w_v of each view are updated using Eqs. (8.4) and (8.5), respectively with $v = 1, \dots, V + 1$.

The proposed MCSRGL method is shown in **Algorithm 1**. After \mathbf{H} is estimated, the cluster index of the sample \mathbf{x}_i is given by the column index corresponding to the maximum value in the i -th row of \mathbf{H} .

Algorithm 1 MCSRGL	
Input:	Data matrix $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$. The similarity matrix \mathbf{S}^v for each view, $v = 1, \dots, V$. Parameters p and λ .
Output:	The consensus cluster label matrix \mathbf{H} . The spectral representation matrix \mathbf{P}^v for each view.
Initialization:	The weights $w_v = \frac{1}{V}$ and $\delta_v = 1$. Initialize \mathbf{S}^v and \mathbf{P}^v as mentioned in section 3.2., $v=1, \dots, V$.
	Initialize \mathbf{S}^{V+1} and \mathbf{P}^{V+1}: Update \mathbf{H} using Eq. (8.9). Estimate \mathbf{S}^{V+1} using Eq. (8.7). Set \mathbf{P}^{V+1} to the C eigenvectors of \mathbf{S}^{V+1} .
	Repeat 1- Update \mathbf{P}^v , $v = 1, \dots, V + 1$ using Eq. (8.11). 2- Update \mathbf{H} using Eq. (8.9). 3- Update \mathbf{S}^{V+1} using Eq. (8.7). 4- Update δ_v , $v = 1, \dots, V + 1$ using Eq. (8.4). 5- Update w_v , $v = 1, \dots, V + 1$ using Eq. (8.5). End

8.1.3 Convergence analysis

In this section, we present the convergence proof of the objective function of **Algorithm 1** according to the weight parameter w_v .

The same procedure can be considered for the weight parameter δ_v . An important Lemna, used to prove the convergence, is introduced below.

Lemma 1. Given two positive constants a and b , we have the following inequality:

$$\sqrt{a} - \frac{a}{2\sqrt{b}} \leq \sqrt{b} - \frac{b}{2\sqrt{a}}. \quad (8.12)$$

Proof of Lemma1:

$$(\sqrt{a} - \sqrt{b})^2 \geq 0 \implies (\sqrt{a})^2 - 2\sqrt{a}\sqrt{b} + (\sqrt{b})^2 \geq 0 \implies 2\sqrt{a}\sqrt{b} - (\sqrt{a})^2 \leq 2(\sqrt{b})^2 - (\sqrt{b})^2 \implies (8.12).$$

Considering the following equation:

$$f(\mathbf{H}) = \sum_{v=1}^{V+1} \delta_v \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2^2.$$

Without loss of any generality, we assume that λ is equal to one in (8.8). By plugging the above definitions for $f(\mathbf{H})$ in problem (8.8), the latter will become:

$$\min \sum_{v=1}^{V+1} w_v \text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H}) + f(\mathbf{H}). \quad (8.13)$$

Suppose that $\hat{\mathbf{H}}$ is the solution of the second step in **Algorithm 1**, and \mathbf{H} is the solution of the objective function, obtained at the previous iteration. We have to prove the following inequality:

$$\sum_{v=1}^{V+1} \sqrt{\text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})} + f(\hat{\mathbf{H}}) \leq \sum_{v=1}^{V+1} \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})} + f(\mathbf{H}). \quad (8.14)$$

It is clear that if we prove the inequality (8.14) for any two consecutive iterations of the Algorithm 1, the convergence of the objective function of the original problem (8.3) will be therefore satisfied, since it is non-increasing over the iterations because each iteration is based on successive minimization steps.

Proof of (8.14):

$\hat{\mathbf{H}}$ is the solution of the second step in **Algorithm 1** (current iteration), and \mathbf{H} is the solution of the objective function, obtained at the previous iteration of **Algorithm 1**. This yields the following:

$$\sum_{v=1}^{V+1} w_v \text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}}) + f(\hat{\mathbf{H}}) \leq \sum_{v=1}^{V+1} w_v \text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H}) + f(\mathbf{H}). \quad (8.15)$$

By plugging the expression of w_v (i.e., Eq. (8.5)), into Eq. (8.15), we get the following inequality:

$$\sum_{v=1}^{V+1} \frac{\text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})}{2 * \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} + f(\hat{\mathbf{H}}) \leq \sum_{v=1}^{V+1} \frac{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}{2 * \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} + f(\mathbf{H}). \quad (8.16)$$

Let $a = \text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})$ and $b = \text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})$. According to **Lemma 1**, we have the following inequality:

$$\sum_{v=1}^{V+1} \left\{ \sqrt{\text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})} - \frac{\text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})}{2 * \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} \right\} \leq \sum_{v=1}^{V+1} \left\{ \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})} - \frac{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}{2 * \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})}} \right\}. \quad (8.17)$$

Finally, we add the left and right sides in Equations (8.16) and (8.17), and we get:

$$\sum_{v=1}^{V+1} \sqrt{\text{Tr} (\hat{\mathbf{H}}^T \mathbf{L}^v \hat{\mathbf{H}})} + f(\hat{\mathbf{H}}) \leq \sum_{v=1}^{V+1} \sqrt{\text{Tr} (\mathbf{H}^T \mathbf{L}^v \mathbf{H})} + f(\mathbf{H}),$$

which proves the convergence of **Algorithm 1**.

8.2 Performance Evaluation

8.2.1 Experimental setup

In this section, we study the clustering performance of the proposed method on six real datasets. We also used the MNIST-10000 and the MNIST-25000 datasets, which can be considered as large datasets. The clustering performance of our method is compared with that of several state-of-the-art methods, including: AWP [20], MLAN [44], SwMC [95], AMGL [13], AASC [96], MVGL [15], CorSC [38], CotSC [34], MVCSK [24], NESE [11], MCLES [67], and SC-Best which is the spectral clustering algorithm of the best view of the data. In our proposed method, two parameters are used: p , which represents the number of most similar sample in the label space for a given sample (used to build the label graph), and λ , the regularization parameter. The value of p varies in the range [5, 25]. Also, the parameter λ varies in the range [10^{-10} , 10^{+2}].

8.2.2 Experimental results

Table 8.1 shows the results obtained using our proposed clustering method and several state-of-the-art methods with the COIL20, ORL, Out-Scene, and NUS datasets. The number in parentheses represents the standard deviation of the indicator value obtained in multiple trials. These results show that for most of these datasets, the best clustering methods are MVCSK, NESE, MCLES, and MCSRGL (proposed method). Therefore, we use these methods for comparison for the large datasets, namely MNIST-10000 and MNIST-25000. Table 8.2 shows the comparison between our method and the state-of-the-art methods: MVCSK, NESE, and MCLES applied to the MNIST-10000 and MNIST-25000 datasets.

8.2.3 Ablation study and parameter sensitivity

In the proposed approach, two parameters are used: λ and p . Since p is used to select the most similar instances in the label space, our conducted experiments show that the best choice of this parameter is 8 for the ORL, Out-Scene, NUS, MNIST-10000, and MNIST-25000 datasets, and it is 25 for the COIL20 dataset. The two subfigures of Figure 8.2 show the clustering performance indicators ACC and NMI as a function of parameter λ for COIL20 and ORL datasets. According to Figure 8.2, the best values of ACC and NMI are obtained for a value of λ equal to 10 for COIL20 and for a value of λ less than 10^{-2} for the ORL dataset.

An ablation study is performed in Table 8.3. Two different variants of MCSRGL are defined: MCSRGL-S and MCSRGL-L. The first variant studies the effect of adding only the term representing

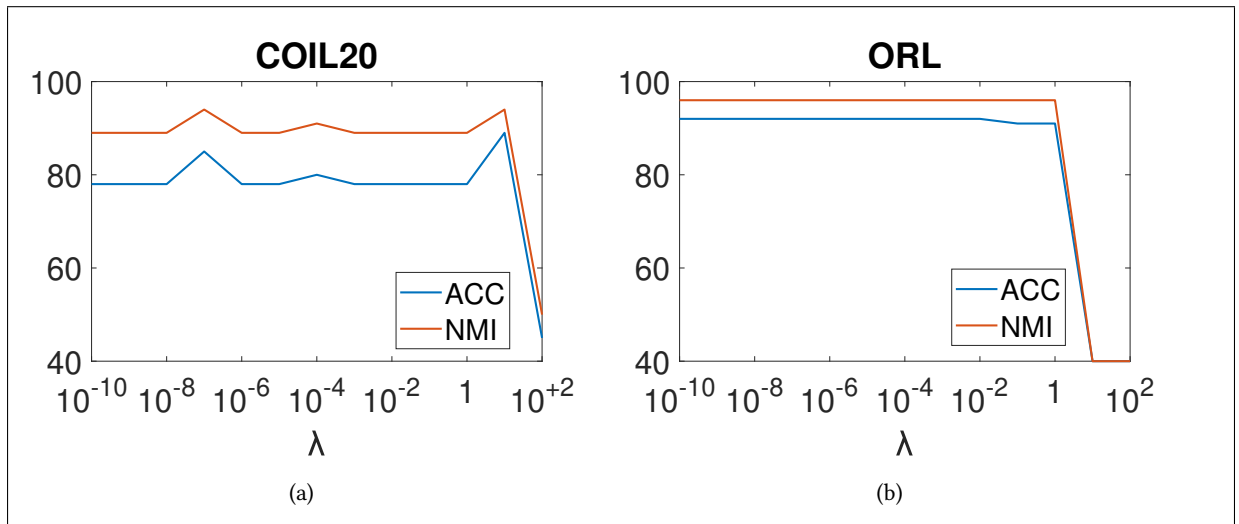
Table 8.1: Clustering performance on the COIL20, ORL, Out-Scene, and NUS datasets.

Dataset	Method	ACC	NMI	Purity	ARI
COIL20	SC-Best	0.73 (\pm 0.01)	0.82 (\pm 0.01)	0.75 (\pm 0.01)	0.68 (\pm 0.02)
	AWP	0.68 (\pm 0.00)	0.87 (\pm 0.00)	0.75 (\pm 0.00)	0.71 (\pm 0.00)
	MLAN	0.84 (\pm 0.00)	0.92 (\pm 0.00)	0.88 (\pm 0.00)	0.81 (\pm 0.00)
	SwMC	0.86 (\pm 0.00)	0.94 (\pm 0.00)	0.90 (\pm 0.00)	0.84 (\pm 0.00)
	AMGL	0.80 (\pm 0.04)	0.91 (\pm 0.02)	0.85 (\pm 0.03)	0.74 (\pm 0.07)
	AASC	0.79 (\pm 0.00)	0.89 (\pm 0.00)	0.83 (\pm 0.00)	0.76 (\pm 0.00)
	MVGL	0.78 (\pm 0.00)	0.88 (\pm 0.00)	0.81 (\pm 0.00)	0.75 (\pm 0.00)
	CorSC	0.68 (\pm 0.04)	0.78 (\pm 0.02)	0.70 (\pm 0.03)	0.62 (\pm 0.03)
	CotSC	0.70 (\pm 0.03)	0.80 (\pm 0.02)	0.72 (\pm 0.03)	0.65 (\pm 0.03)
	MVCSK	0.65 (\pm 0.04)	0.80 (\pm 0.02)	0.70 (\pm 0.03)	0.61 (\pm 0.05)
	NESE	0.77 (\pm 0.00)	0.88 (\pm 0.00)	0.82 (\pm 0.00)	0.69 (\pm 0.00)
	MCLES	0.79 (\pm 0.00)	0.88 (\pm 0.00)	0.83 (\pm 0.00)	0.75 (\pm 0.00)
	MCSRGL	0.89 (\pm 0.00)	0.94 (\pm 0.00)	0.89 (\pm 0.00)	0.84 (\pm 0.00)
	ORL	SC-Best	0.66 (\pm 0.02)	0.76 (\pm 0.02)	0.71 (\pm 0.02)
AWP		0.80 (\pm 0.00)	0.91 (\pm 0.00)	0.83 (\pm 0.00)	0.76 (\pm 0.00)
MLAN		0.78 (\pm 0.00)	0.88 (\pm 0.00)	0.82 (\pm 0.00)	0.67 (\pm 0.00)
SwMC		0.77 (\pm 0.00)	0.90 (\pm 0.00)	0.83 (\pm 0.00)	0.62 (\pm 0.00)
AMGL		0.75 (\pm 0.02)	0.90 (\pm 0.02)	0.82 (\pm 0.02)	0.63 (\pm 0.09)
AASC		0.82 (\pm 0.02)	0.91 (\pm 0.01)	0.85 (\pm 0.01)	0.76 (\pm 0.02)
MVGL		0.75 (\pm 0.00)	0.88 (\pm 0.00)	0.80 (\pm 0.00)	0.55 (\pm 0.00)
CorSC		0.77 (\pm 0.03)	0.90 (\pm 0.01)	0.82 (\pm 0.03)	0.72 (\pm 0.04)
CotSC		0.75 (\pm 0.04)	0.87 (\pm 0.01)	0.78 (\pm 0.03)	0.67 (\pm 0.03)
MVCSK		0.85 (\pm 0.02)	0.94 (\pm 0.01)	0.88 (\pm 0.02)	0.81 (\pm 0.02)
NESE		0.82 (\pm 0.00)	0.91 (\pm 0.00)	0.85 (\pm 0.00)	0.75 (\pm 0.00)
MCLES		0.84 (\pm 0.00)	0.94 (\pm 0.00)	0.88 (\pm 0.00)	0.79 (\pm 0.00)
MCSRGL		0.92 (\pm 0.00)	0.96 (\pm 0.00)	0.93 (\pm 0.00)	0.88 (\pm 0.00)
Out-Scene		SC-best	0.47 (\pm 0.01)	0.39 (\pm 0.01)	0.57 (\pm 0.01)
	AWP	0.65 (\pm 0.00)	0.51 (\pm 0.00)	0.65 (\pm 0.00)	0.42 (\pm 0.00)
	MLAN	0.55 (\pm 0.02)	0.47 (\pm 0.01)	0.55 (\pm 0.02)	0.33 (\pm 0.03)
	SwMC	0.50 (\pm 0.00)	0.47 (\pm 0.00)	0.50 (\pm 0.00)	0.38 (\pm 0.00)
	AMGL	0.51 (\pm 0.05)	0.45 (\pm 0.03)	0.52 (\pm 0.04)	0.34 (\pm 0.05)
	AASC	0.60 (\pm 0.00)	0.48 (\pm 0.00)	0.60 (\pm 0.00)	0.35 (\pm 0.00)
	MVGL	0.42 (\pm 0.00)	0.31 (\pm 0.00)	0.43 (\pm 0.00)	0.16 (\pm 0.00)
	CorSC	0.51 (\pm 0.04)	0.39 (\pm 0.03)	0.52 (\pm 0.03)	0.31 (\pm 0.02)
	CotSC	0.38 (\pm 0.02)	0.22 (\pm 0.01)	0.39 (\pm 0.02)	0.16 (\pm 0.01)
	MVCSK	0.65 (\pm 0.01)	0.52 (\pm 0.00)	0.65 (\pm 0.01)	0.42 (\pm 0.00)
	NESE	0.63 (\pm 0.00)	0.53 (\pm 0.00)	0.66 (\pm 0.00)	0.46 (\pm 0.00)
	MCLES	0.65 (\pm 0.00)	0.53 (\pm 0.00)	0.67 (\pm 0.00)	0.46 (\pm 0.00)
	MCSRGL	0.70 (\pm 0.00)	0.55 (\pm 0.00)	0.70 (\pm 0.00)	0.47 (\pm 0.00)
	NUS	SC-Best	0.21(\pm 0.01)	0.09(\pm 0.01)	0.21(\pm 0.01)
AWP		0.28(\pm 0.00)	0.15(\pm 0.00)	0.29(\pm 0.00)	0.09(\pm 0.00)
MLAN		0.25(\pm 0.00)	0.15(\pm 0.00)	0.26(\pm 0.00)	0.04(\pm 0.00)
SwMC		0.15(\pm 0.00)	0.08(\pm 0.00)	0.17(\pm 0.00)	0.01(\pm 0.00)
AMGL		0.25(\pm 0.01)	0.13(\pm 0.01)	0.27(\pm 0.01)	0.07(\pm 0.01)
AASC		0.25(\pm 0.00)	0.13(\pm 0.00)	0.27(\pm 0.00)	0.06(\pm 0.00)
MVGL		0.15(\pm 0.00)	0.07(\pm 0.00)	0.16(\pm 0.00)	0.01(\pm 0.00)
CorSC		0.27(\pm 0.01)	0.14(\pm 0.01)	0.29(\pm 0.01)	0.09(\pm 0.01)
CotSC		0.29(\pm 0.01)	0.16(\pm 0.01)	0.30(\pm 0.01)	0.09(\pm 0.01)
MVCSK		0.26(\pm 0.01)	0.15(\pm 0.00)	0.28(\pm 0.00)	0.08(\pm 0.00)
NESE		0.30(\pm 0.00)	0.17(\pm 0.00)	0.32(\pm 0.00)	0.10(\pm 0.00)
MCLES		0.30(\pm 0.00)	0.16(\pm 0.00)	0.32(\pm 0.00)	0.10(\pm 0.00)
MCSRGL		0.33(\pm 0.00)	0.18(\pm 0.00)	0.35(\pm 0.00)	0.11(\pm 0.00)

the smoothness of the cluster labels over all graphs with the objective function of NESE. The resulting method is called MCSRGL-S. The second variant considers the effect of including the label graph as an additional view in the objective function of NESE. The resulting method is called MCSRGL-L. The results obtained with NESE, MCSRGL-S, MCSRGL-L and MCSRGL are summarized in Table 8.3. We used two datasets: ORL and Out-Scene. From the results in Table

Table 8.2: Clustering performance on the MNIST-10000 and the MNIST-25000 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
MNIST-10000	MVCSK	0.49 (± 0.00)	0.41 (± 0.00)	0.50 (± 0.00)	0.29 (± 0.00)
	NESE	0.81 (± 0.00)	0.83 (± 0.00)	0.85 (± 0.00)	0.76 (± 0.00)
	MCLES	0.80 (± 0.00)	0.83 (± 0.00)	0.85 (± 0.00)	0.77 (± 0.00)
	MCSRGL	0.81 (± 0.00)	0.84 (± 0.00)	0.86 (± 0.00)	0.78 (± 0.00)
MNIST-25000	MVCSK	0.47 (± 0.00)	0.38 (± 0.00)	0.52 (± 0.00)	0.25 (± 0.00)
	NESE	0.72 (± 0.00)	0.75 (± 0.00)	0.77 (± 0.00)	0.65 (± 0.00)
	MCLES	0.73 (± 0.00)	0.76 (± 0.00)	0.78 (± 0.00)	0.64 (± 0.00)
	MCSRGL	0.77 (± 0.00)	0.80 (± 0.00)	0.82 (± 0.00)	0.69 (± 0.00)

**Figure 8.2:** Clustering performance ACC (%) and NMI (%) as a function of λ on the COIL20 and ORL datasets.

8.3, we can see that both the inclusion of the label space and the smoothness term contributed to the good clustering performance of our method. We can also see that the label graph contributed more than the smoothness term.

Table 8.3: Ablation study with different conditions.

Dataset	Variant	ACC	NMI	Purity	ARI
ORL	NESE	0.82	0.91	0.85	0.75
	MCSRGL-S	0.83	0.91	0.86	0.76
	MCSRGL-L	0.92	0.96	0.93	0.88
	MCSRGL	0.92	0.96	0.93	0.88
Out-Scene	NESE	0.63	0.53	0.66	0.46
	MCSRGL-S	0.63	0.53	0.66	0.47
	MCSRGL-L	0.69	0.54	0.69	0.46
	MCSRGL	0.70	0.55	0.70	0.47

8.2.4 Analysis of results and method comparison

The performance of some state-of-the-art methods and our proposed method are shown in Tables 8.1 and 8.2. First, it is noticeable that in Table 8.1, the result of "SC-Best", which represents the

best result obtained by applying the spectral clustering algorithm to each view, is lower than the results obtained by most multi-view clustering algorithms in most cases.

From Table 8.1, it can be seen that our algorithm achieves high performance on all datasets. More specifically, MCSRGL achieves high clustering results compared to the NESE method. For the COIL20 dataset, when we compare the result of our method with the SwMC method, we find that the result for the ACC indicator is higher than that of SwMC, for the NMI and ARI indicators it is the same as that of SwMC, and for the purity indicator the result of our method is only 1% lower than that obtained by SwMC. However, when we compare this result with the method NESE method (our method is an improved version of this last method) and with all other methods in Table 8.1, we find that MCSRGL is the most superior, which justifies the superiority of this method on the COIL20 dataset. Moreover, Table 8.1 shows that both methods NESE and MVCSK are more efficient than the other methods, which leads us to use them as comparison methods on the large MNIST-10000 dataset.

Table 8.2 shows the result of MCSRGL on the MNIST-10000 and MNIST-25000 datasets. This result is similar to that of NESE for the indicator ACC, and higher than that of NESE for the indicators NMI, purity, and ARI, for the MNIST-10000 dataset, however, according to the MNIST-25000, our algorithm gives higher performance than the other three methods, indicating that our method performs well once applied to large datasets.

All these results show that the added constraint on the cluster label matrix and the use of the additional label graph contribute to better performance.

With only one parameter, our proposed method is very practical. It achieves a good improvement of all cluster evaluation metrics. The average of the improvement of each cluster evaluation metric is calculated by taking the average of the difference between our method and the most competitive method across all datasets. The maximum improvement of each cluster indicator metric is also recorded. For the Accuracy indicator, the average improvement is about 4 % and the maximum improvement obtained for the ORL dataset is 7 %. For the Normalized Mutual Information indicator, the average improvement is 2 % and the maximum improvement is 4 % for the MNIST-25000 dataset. In addition, the Purity indicator has been improved by 3 % on average and the maximum improvement is 5 % for the ORL dataset. The average improvement of the adjusted rand index is about 2 % and the maximum improvement of this indicator is 7 % for the dataset ORL.

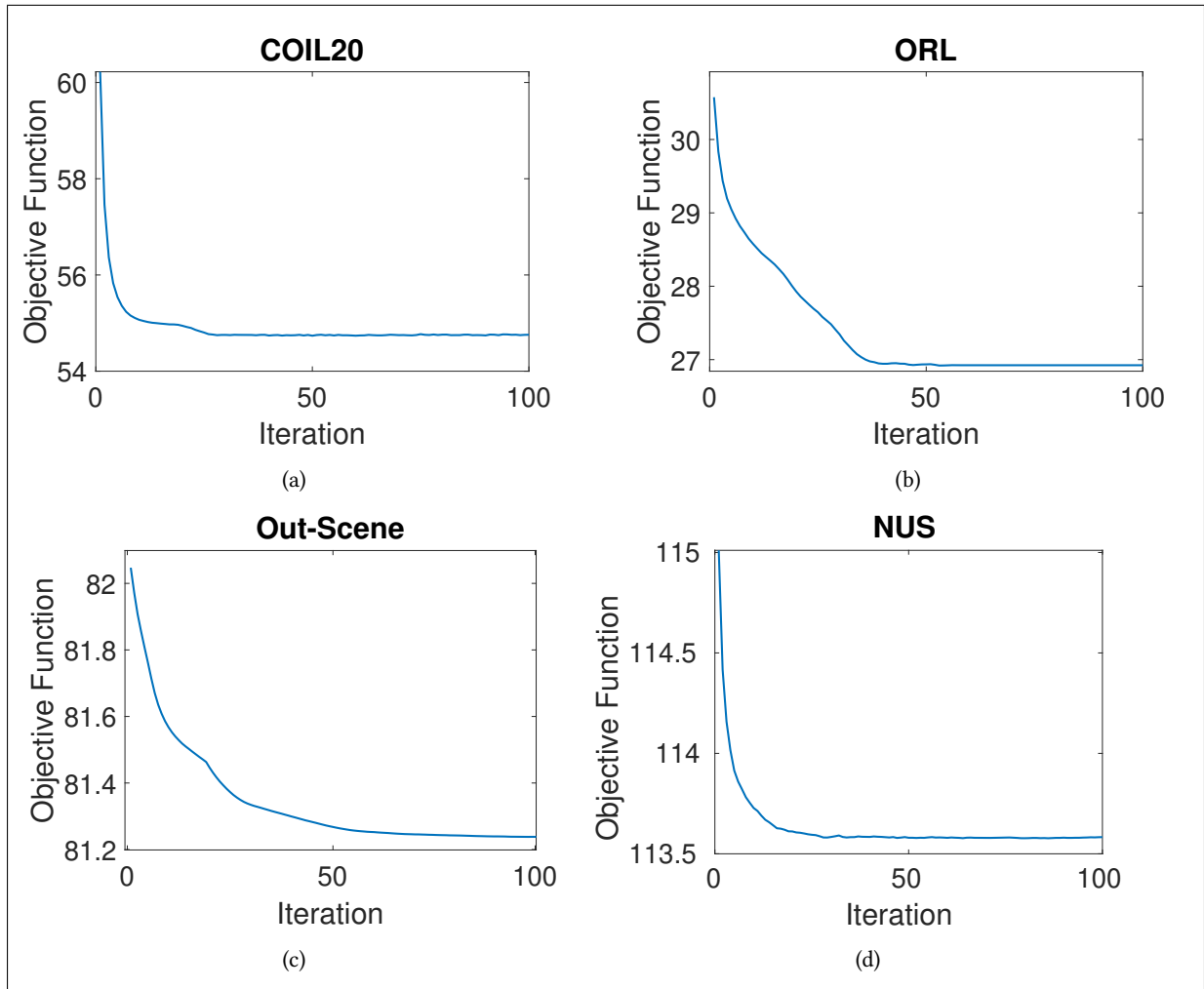


Figure 8.3: Convergence of MCSRGL on the COIL20, ORL, Out-Scene, and NUS datasets.

8.2.5 Convergence study

In this section, the convergence of MCSRGL on four datasets is shown in Figure 8.3. In our method, the number of iterations is set to 100. For each iteration, the value of the objective function is calculated using the objective function given by Eq. (8.8). Figure 8.3 shows the convergence of our method for the COIL20, ORL, Out-Scene, and NUS datasets. As can be seen from Figure 8.3, the convergence of our proposed method is fast. The solution was obtained in less than 40 iterations.

8.2.6 Clustering visualization

To make it more intuitive, we visualize the clustering obtained by MVCSK, NESE and MCSRGL methods on two datasets: COIL20, and ORL using the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [98]. The features of a given sample (used as input for t-SNE) are set to the corresponding row in the spectral representation matrix \mathbf{P} for the MVCSK method, and to the

corresponding row in the cluster label matrix \mathbf{H} for both NESE and MCSRGL methods. The color in these subimages corresponds to the estimated clusters, and each point represents one image. It can be clearly seen that the performances in Figure 8.4 are consistent with the quantitative results in Table 8.1. The results from Figure 8.4 also show that our method MCSRGL is able to obtain better separated clusters than the other two methods: NESE and MVCSK, for the two datasets COIL20 and ORL. It also shows that MVCSK has more clustering errors compared to NESE and MCSRGL, which also justifies the result obtained in Table 8.1. This shows that incorporating the label space via an additional graph and adding a cluster label smoothness term improve the performance of MCSRGL compared to the other two methods NESE and MVCSK.

8.3 Conclusion

In this chapter, a novel multi-view graph-based clustering method is proposed. This method is an improvement of the graph-based clustering method presented in [11], and called Nonnegative Embedding and Spectral Embedding (NESE). We introduced a new criterion that uses and exploits two concepts. The first concept is based on adding a constraint on the cluster label matrix given by the smoothness of the cluster labels over all graphs. The second concept is based on integrating the label space via an additional graph into the estimation framework. The impact of these two concepts on the final clustering performance is investigated. Experimental results on real image datasets of different types and sizes show the effectiveness and superiority of the proposed approach.

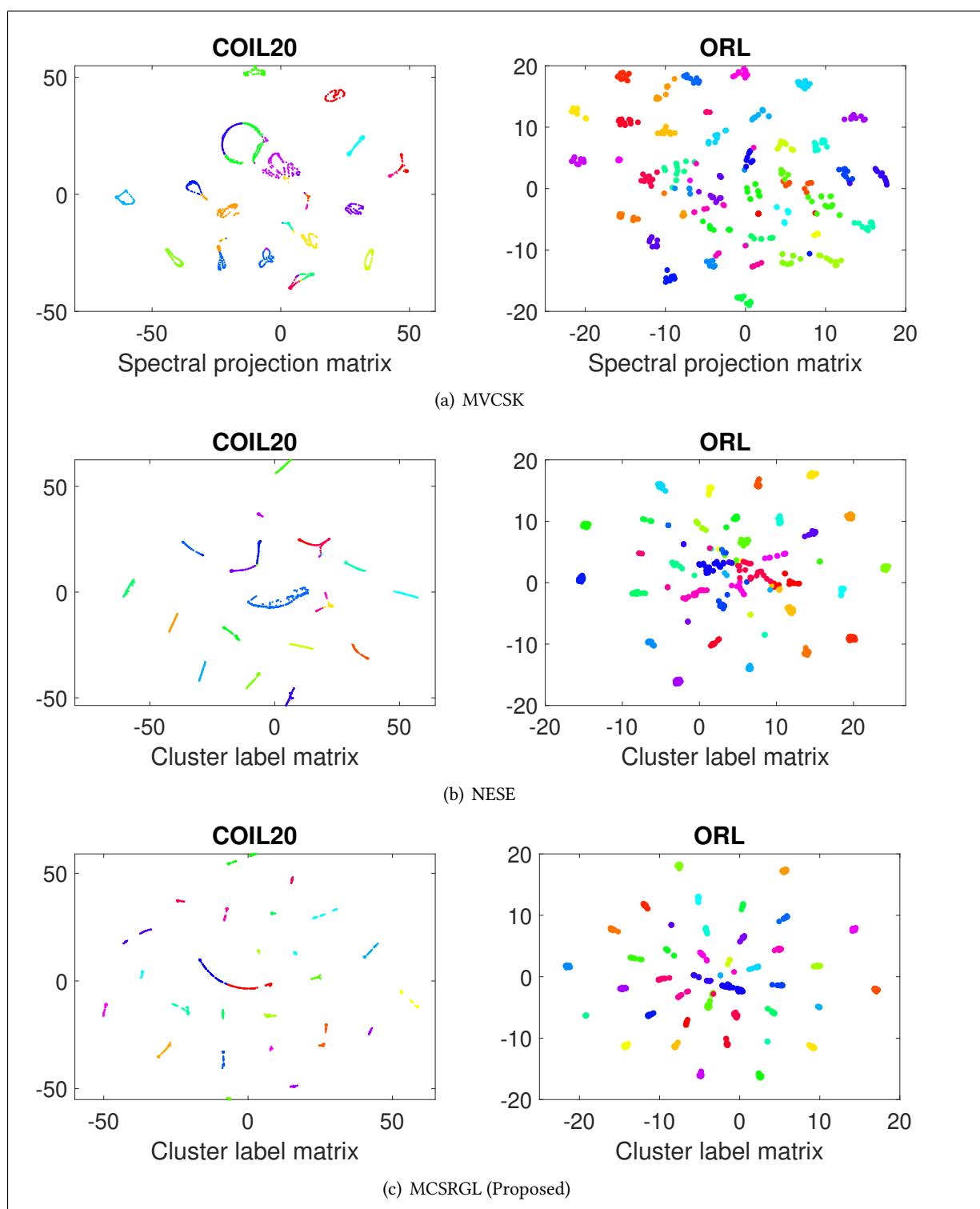


Figure 8.4: t-SNE visualization of the spectral projection, and cluster label matrices on the COIL20, and ORL datasets.

A Unified Framework for Multi-view Clustering via Consensus Graph and Spectral Representation Learning

Despite the high performance of the aforementioned multi-view clustering methods, they directly fuse all similarity matrices of all views. The performance of these algorithms can be affected by noisy affinity matrices. To overcome this drawback, in this chapter we propose a novel method called One Step Multi-view Clustering via Consensus Graph Learning and Soft Clustering Assignments (OSMGSCA). Instead of directly merging the similarity matrices of different views, which may contain noise, a step of learning the individual graphs and the consensus graph is integrated in the new proposed model. In this model, we force the similarity matrices of different views to be too similar, which eliminates the problem of noisy views. Also, similar to the MKGNE and MVCGE methods presented in Chapter 5 and Chapter 6, this novel approach considers learning of the consistent graphs using a kernel representation of the views. Moreover, our model makes it possible to obtain the final clustering assignment directly without another step by using a nonnegative embedding matrix. This approach can solve five subtasks simultaneously. It jointly estimates the joint similarity matrix of all views, the similarity matrix of each view, the corresponding spectral projection matrix, the unified clustering indicator matrix, and automatically gives the weight of each view without using additional parameters. **In addition, another version of our method is also studied in this chapter. This method differs from the first one by using a consensus spectral projection matrix and a consensus Laplacian**

matrix over all views. An iterative algorithm is proposed to solve the optimization problem of these two methods. Our methods are tested on several real datasets, in order to prove their usefulness and their superiority to the state of the art methods. The main contributions of this chapter are given below.

1. We propose an end-to-end solution starting from the data or their kernel representations. The proposed approach can jointly estimate the graph of each view, the consensus graph, the spectral projection matrices for all views, the nonnegative embedding matrix (soft clustering assignments), and the weights of each view. This joint estimation results in friendly clustering entities, namely the graphs and the spectral representations.
2. By using the nonnegative embedding matrix representing the soft clustering assignments, the final clustering assignment is determined directly without any post-processing step.
3. Our method inherits the advantages of consensus multi-view learning methods, matrix factorization methods, and graph-based learning methods.
4. We use an iterative algorithm to solve the resulting optimization problem. Several real-world multi-view datasets of different types and sizes are used to validate the effectiveness of our algorithm against other competing methods.
5. Another variant of our proposed method is also proposed and tested. In this variant, the soft clustering assignments are associated with the consensus graph and spectral projection, while in the first variant they are associated with the view-based graphs and spectral projections.

9.1 Proposed Approach

Inspired by the method in [104], which simultaneously computes the similarity matrix (graph matrix) of each view, the consensus similarity and the final clustering assignment, we develop a new method, namely One Step Multi-view Clustering via Consensus Graph Learning and Soft Clustering Assignments (OSMGSCA). Unlike the method presented in [104], our proposed method ensures that the cluster assignments satisfy graph-based smoothing as well as graph-based convolution of the different spectral representations.

This method can learn the similarity matrices of all views and their corresponding spectral projection matrices, consensus similarity matrix, and final clustering assignment jointly by using

the nonnegative embedding matrix to obtain the cluster assignments of each sample directly without a post-processing step, and it gives the weight of each view automatically without using additional parameters. Moreover, we propose a variant of the proposed method called Unified Multi-view Clustering via Joint Graph Learning and Nonnegative Matrix Assignments (U-MCJGLNMA). In this variant the constraints smoothing and convolution are built on the consensus graph and the consensus spectral representation.

These two methods are investigated in this chapter. We first explain the approach OSMGSCA. Then, we present another variant of our method, which consists in using some unified graph and spectral representation instead of using the matrices of the individual views. This method is referred to as U-MCJGLNMA.

OSMGSCA can estimate four types of entities together: 1) the consensus similarity matrix, 2) the similarity matrix of each view, 3) the corresponding spectral projection matrix of each similarity matrix, and 4) the consensus nonnegative embedding matrix. This method automatically updates the weights of each view without any additional parameter. We will explain the main concepts of our method and then present the global objective function. We will start by presenting the main modules and terms used by OSMGSCA. Finally, we will give the global objection function allowing to estimate the unknowns.

Let n denote the total number of samples. These are represented by V different views or feature vectors. These should be grouped into C disjoint clusters. Given the data matrix of each view, this matrix can be denoted as $\mathbf{X}^v = (\mathbf{x}_1^v, \mathbf{x}_2^v, \dots, \mathbf{x}_n^v) \in \mathbb{R}^{d^v \times n}$, where d^v is the dimension of the feature vector in the view v , where $v = 1, \dots, V$. The kernel trick is used to map our data into a space where they will be better represented. Thus, the kernel matrix of each view is represented by \mathbf{K}^v . Without loss of generality, we will assume that the kernel matrices are obtained using the Gaussian kernel with adaptive scales.

We aim to jointly estimate the following unknown matrices: $\mathbf{S}^* \in \mathbb{R}^{n \times n}$, $\mathbf{S}^v \in \mathbb{R}^{n \times n}$, $\mathbf{P}^v \in \mathbb{R}^{n \times C}$, and $\mathbf{H} \in \mathbb{R}^{n \times C}$. Our proposed criterion is composed by five terms. First, the self expressive property and the kernel matrices are used to estimate the similarity graph of each view

The first and second terms will be given by the equation below.

$$\min_{\mathbf{S}^v, v=1, \dots, V} \sum_{v=1}^V Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S}^v + \mathbf{S}^{vT} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 \quad s.t. \quad 0 \leq \mathbf{S}^v \leq \mathbf{1}, \mathbf{S}^v \mathbf{1} = \mathbf{1}, \text{diag}(\mathbf{S}^v) = \mathbf{0}. \quad (9.1)$$

The second term is a simple regularization that provides stable solutions. As it is known, multi-view clustering consists of assigning each sample to the same group in different views, i.e. the similarity

between two points must be roughly the same in different views. So each similarity matrix \mathbf{S}^v will be enforced to align with a consensus similarity matrix \mathbf{S}^* to learn different information from multiple views. This matrix explores the complementary of different views by the additional information given by each similarity matrix. Besides, the effect of noise and outliers present in a certain view will be reduced. Therefore, instead of doing similarity fusion, we impose that the individual similarities matrices \mathbf{S}^v are close to a consensus similarity matrix \mathbf{S}^* . Thus, the third term of our objective function will be given by:

$$\min_{\mathbf{S}^*} \sum_{v=1}^V \|\mathbf{S}^* - \mathbf{S}^v\|_F^2. \quad (9.2)$$

As we have mentioned before, we aim to directly obtain the cluster assignments represented by a nonnegative embedding matrix \mathbf{H} . In our work, we impose that this matrix is a kind of convolution of each view spectral representation \mathbf{P}^v over its corresponding graph \mathbf{S}^v . The index of the highest value in the row vector \mathbf{H}_{i*} indicates the cluster to which the data sample belongs. Then the fourth term of our objective function, which is inspired by NESE [11], is given by:

$$\min_{\mathbf{H}, \mathbf{P}^v} \sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_F \quad s.t. \ , \ \mathbf{H}^T \mathbf{H} = \mathbf{I}, \ \mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}. \quad (9.3)$$

To get more accurate nonnegative embedding matrix, a view-based cluster label smoothness term over the matrix \mathbf{H} is added in the objective function of our method, and it is given by:

$$\min_{\mathbf{H}} \sum_{v=1}^V \left(\frac{1}{2} \sum_i \sum_j \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2 S_{ij}^v \right)^{1/2} = \min_{\mathbf{H}} \sum_{v=1}^V \sqrt{\text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H})}, \quad (9.4)$$

where $\mathbf{L}^v = \mathbf{D}^v - \mathbf{S}^v \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the similarity matrix of each view v , and \mathbf{D}^v is the diagonal matrix of the v -th similarity matrix with its elements are given by: $D_{ii}^v = \sum_{j=1}^n \frac{S_{ij}^v + S_{ji}^v}{2}$. This term indicates that if the value of the similarity matrix corresponding to the similarity between two data points \mathbf{x}_i^v and \mathbf{x}_j^v is large, the value of \mathbf{H}_{i*} will be very close to \mathbf{H}_{j*} , and thus \mathbf{x}_i^v and \mathbf{x}_j^v will be forced to be in the same cluster.

Two view's weights w_v and δ_v are used by our method to automatically allocate the weights of all views. They are given by:

$$w_v = \frac{1}{2 \sqrt{\text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H})}}. \quad (9.5)$$

$$\delta_v = \frac{1}{2 * \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2}. \quad (9.6)$$

Therefore, our learning model is obtained by combining all terms. Thus, the final objective function will be given by Eq. (9.7).

$$\begin{aligned} \min_{\mathbf{S}^v, \mathbf{P}^v, \mathbf{S}^*, \mathbf{H}} \sum_{v=1}^V \{ & Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S}^v + \mathbf{S}^{vT} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 + \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2 + \lambda_2 \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2 \\ & + \lambda_3 w_v Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H}), \end{aligned} \quad (9.7)$$

where λ_1 , λ_2 , and λ_3 are regularization parameters.

In Eq.(9.7), we used the following two facts: (1) minimizing $\sum_{v=1}^V \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_F$ is equivalent to minimizing $\sum_{v=1}^V \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2$, and (2) minimizing $\sum_{v=1}^V \sqrt{Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H})}$ is equivalent to minimizing $\sum_{v=1}^V w_v Tr(\mathbf{H}^T \mathbf{L}^v \mathbf{H})$.

The illustration of the proposed method is shown in Figure 9.1.

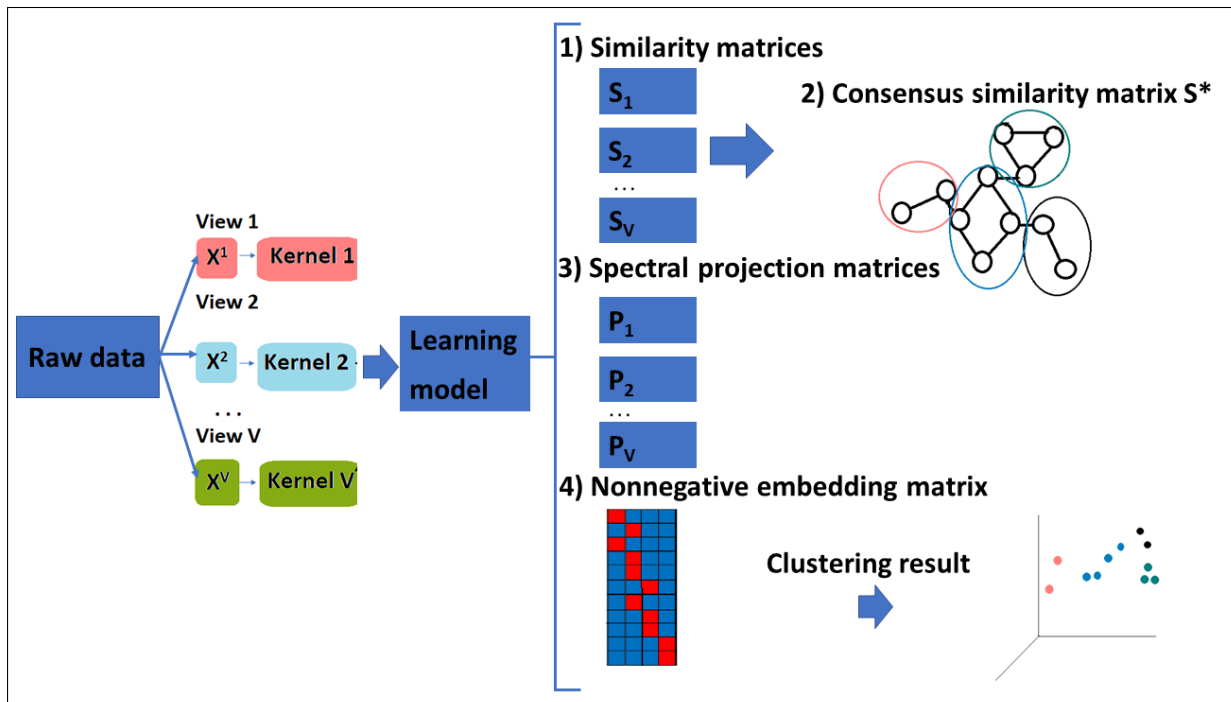


Figure 9.1: Illustration of the OSMGSCA method.

In the above criterion, the convolution and smoothness of \mathbf{H} is performed on the individual graphs \mathbf{S}^v , and spectral representations \mathbf{P}^v .

Therefore, a variant of the proposed approach can perform these tasks on the consensus graph \mathbf{S}^* and a consensus spectral representation \mathbf{P}^* . This gives rise to the version of our method U-MCJGLNMA. The difference between OSMGSCA and U-MCJGLNMA is that the latest consists of using the unified matrices in the last two terms instead of using the matrices of all views. The

objective function of this method will be given in Eq. (9.8).

$$\min_{\mathbf{S}^v, \mathbf{S}^*, \mathbf{P}^*, \mathbf{H}} \sum_{v=1}^V \{Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S}^v + \mathbf{S}^{vT} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 + \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2\} + \lambda_2 \|\mathbf{S}^* - \mathbf{H} \mathbf{P}^{*T}\|_2^2 + \lambda_3 Tr(\mathbf{H}^T \mathbf{L}^* \mathbf{H}). \quad (9.8)$$

The illustration of the proposed U-MCJGLNMA method is shown in Figure 9.2.

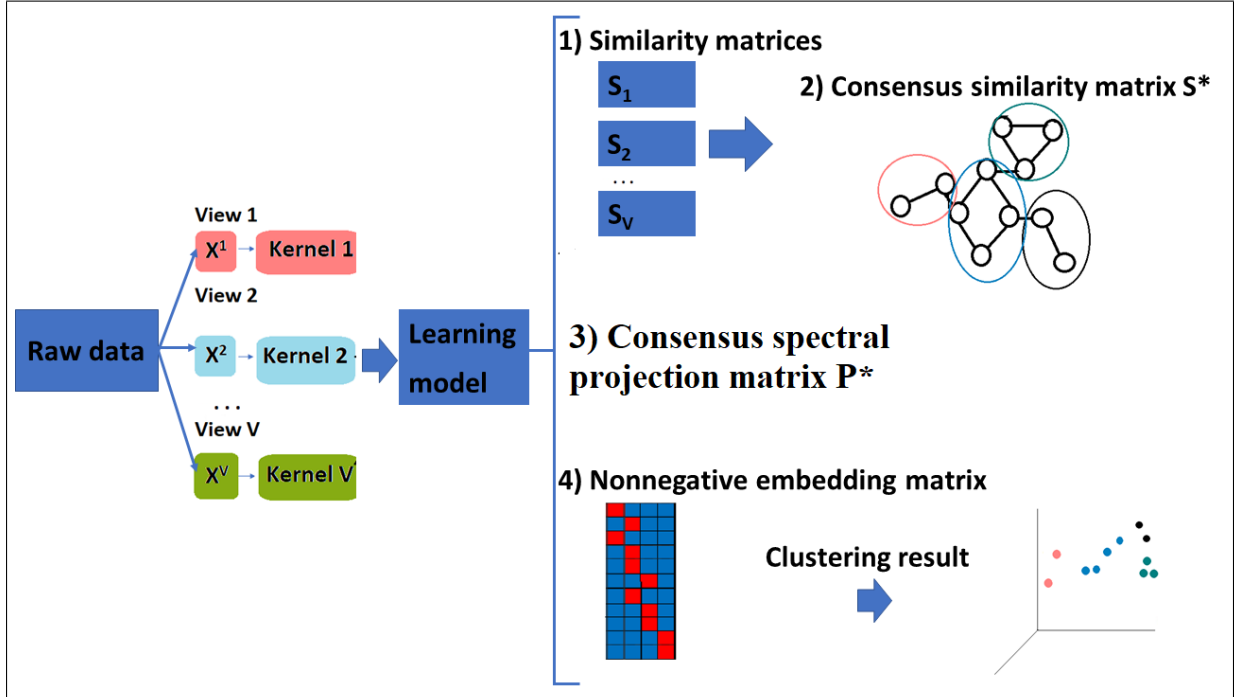


Figure 9.2: Illustration of the U-MCJGLNMA method.

9.2 Optimization of the model

9.2.1 Optimization of OSMGSCA (Eq. (9.7))

In this section, the optimization scheme of the objective function in (9.7) is introduced. Eq. (9.7) is not jointly convex with respect to the unknown matrices. However, it is convex with respect to each of them while holding the other fixed. Therefore, (9.7) can be effectively solved with an alternating optimization algorithm. First, the optimization procedure of OSMGSCA is introduced to update the matrices \mathbf{S}^v , \mathbf{S}^* , \mathbf{P}^v , and \mathbf{H} .

Besides, we initialize the matrices \mathbf{S}^v and \mathbf{P}^v by using the same method as in [93], and the matrix \mathbf{S}^* is initialized by taking the average of the matrices \mathbf{S}^v .

Then, by using the iterative procedure, the algorithm performs the updating steps as follows.

Update H: Fixing \mathbf{S}^* , \mathbf{S}^v , \mathbf{P}^v , w_v and δ_v , the nonnegative embedding matrix \mathbf{H} is updated by computing the derivative of the functional in (9.7) with respect to \mathbf{H} :

$$\frac{\partial f}{\partial \mathbf{H}} = \sum_{v=1}^V 2 \delta_v \lambda_2 (\mathbf{H} - \mathbf{S}^v \mathbf{P}^v) + 2 \lambda_3 \sum_{v=1}^V w_v \mathbf{L}^v \mathbf{H}.$$

To get the optimal solution \mathbf{H} , this derivative is vanished. After some simple algebraic manipulations, we can get the \mathbf{H} :

$$\mathbf{H} = \left(\sum_{v=1}^V (\delta_v \lambda_2 \mathbf{I} + \lambda_3 w_v \mathbf{L}^v) \right)^{-1} \left(\sum_{v=1}^V \delta_v \lambda_2 \mathbf{S}^v \mathbf{P}^v \right). \quad (9.9)$$

Thus, to satisfy the orthogonality constraint imposed on the matrix \mathbf{H} , we carry out an orthogonalization for the matrix \mathbf{H} obtained by Equation (9.9). In addition, to satisfy the constraint of positivity imposed on the matrix \mathbf{H} , we use the element-wise ReLU (Rectified Linear Unit) operator to the elements of the matrix \mathbf{H} .

Update \mathbf{P}^v : Fixing \mathbf{S}^* , \mathbf{S}^v , \mathbf{H} , w_v and δ_v , the objective function of our method will be equal to:

$$\min_{\mathbf{P}^v} \sum_{v=1}^V \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2. \quad (9.10)$$

We update \mathbf{P}^v , $v = 1, \dots, V$ separately. Knowing that $\mathbf{P}^{vT} \mathbf{P}^v = \mathbf{I}$, the above problem is the famous orthogonal Procrustes problem. To get the solution of this problem, we use the singular value decomposition of $\mathbf{S}^{vT} \mathbf{H}$. Let $\mathbf{U}^v \Sigma^v \mathbf{V}^{vT} = \text{SVD}(\mathbf{S}^{vT} \mathbf{H})$. Thus, the solution of Equation (9.10) is given by:

$$\mathbf{P}^v = \mathbf{U}^v \mathbf{V}^{vT} \quad \text{with} \quad \mathbf{U}^v \Sigma^v \mathbf{V}^{vT} = \text{SVD}(\mathbf{S}^{vT} \mathbf{H}). \quad (9.11)$$

Update \mathbf{S}^v : Fixing \mathbf{S}^* , \mathbf{P}^v , \mathbf{H} , w_v and δ_v , we have to solve the following problem.

$$\begin{aligned} \min_{\mathbf{S}^v} \sum_{v=1}^V \text{Tr}(\mathbf{K}^v - 2 \mathbf{K}^v \mathbf{S}^v + \mathbf{S}^{vT} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 + \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2 + \lambda_2 \delta_v \|\mathbf{S}^v - \mathbf{H} \mathbf{P}^{vT}\|_2^2 \\ + \lambda_3 \mathbf{w}^v \text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H}). \end{aligned} \quad (9.12)$$

The following identity is a well-known identity derived from the spectral clustering analysis:

$$\text{Tr}(\mathbf{H}^T \mathbf{L}^v \mathbf{H}) = \frac{1}{2} \sum_i \sum_j \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2 S_{ij}^v = \text{Tr}(\mathbf{Q} \mathbf{S}^v), \quad (9.13)$$

where \mathbf{H}_{i*} and \mathbf{H}_{j*} are the i -th and j -th rows of the matrix \mathbf{H} . The matrix \mathbf{Q} is given by: $Q_{ij} = \frac{1}{2} \|\mathbf{H}_{i*} - \mathbf{H}_{j*}\|^2$. If we replace \mathbf{Q} by its expression in Eq. (9.12), we obtain:

$$\begin{aligned}
 \min_{\mathbf{S}^v} \sum_{v=1}^V Tr(\mathbf{K}^v - 2\mathbf{K}^v \mathbf{S}^v + \mathbf{S}^{vT} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 + \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2 + \lambda_2 \delta_v \|\mathbf{S}^v - \mathbf{H}\mathbf{P}^{vT}\|_2^2 \\
 + \lambda_3 \mathbf{w}^v Tr(\mathbf{Q}\mathbf{S}^v) s.t. \quad 0 \leq \mathbf{S}^v \leq \mathbf{1}, \mathbf{S}^{vT} \mathbf{1} = \mathbf{1}, diag(\mathbf{S}^v) = 0.
 \end{aligned} \tag{9.14}$$

We update \mathbf{S}^v , $v = 1, \dots, V$ separately. For each view, we first calculate \mathbf{S}^v without the constraints. Then, the obtained solution is projected in the constrained space.

By setting the derivative of this objective function to zero, we obtain the expression of $\hat{\mathbf{S}}^v$, which represents the similarity matrix for each view without taking into account the constraints:

$$\hat{\mathbf{S}}^v = (\mathbf{K}^v + (1 + \lambda_1 + \lambda_2 \delta_v) \mathbf{I})^{-1} (\mathbf{K}^v + \lambda_1 \mathbf{S}^* + \lambda_2 \delta_v \mathbf{H}\mathbf{P}^{vT} - \frac{1}{2} w_v \lambda_3 \mathbf{Q}) \tag{9.15}$$

Then, to satisfy the constraints over each similarity matrix, we project it into a constrained space. Moreover, for a fixed v , the optimization of every row $\mathbf{S}_{i,:}^v$ is independent on the optimization of the other rows.

Therefore, the following problem will be the solution of \mathbf{S}^v : For each view v and for each row of the similarity matrix, we get the following minimization problem:

$$\min_{0 \leq \mathbf{S}_{i,:}^v \leq \mathbf{1}, \mathbf{S}_{i,:}^{vT} \mathbf{1} = 1, S_{i,i}^v = 0} \|\mathbf{S}_{i,:}^v - \hat{\mathbf{S}}_{i,:}^v\|_2^2.$$

The Lagrange function of the above minimization problem can be written as:

$$\mathcal{L}(\mathbf{S}_{i,:}^v, \alpha_i^v, \beta_i) = \|\mathbf{S}_{i,:}^v - \hat{\mathbf{S}}_{i,:}^v\|_2^2 - \alpha_i^v (\mathbf{S}_{i,:}^{vT} \mathbf{1} - 1) - \beta_i^T \mathbf{S}_{i,:}^v, \tag{9.16}$$

where α_i^v and $\beta_i^v \geq \mathbf{0}$ are Lagrangian multipliers. Therefore, the i -th row of the similarity matrix for each view \mathbf{S}^v is equal to [105, 55]:

$$\mathbf{S}_{i,:}^v = \max(\hat{\mathbf{S}}_{i,:}^v + \alpha_i^v \mathbf{1}^T, 0), S_{i,i}^v = 0. \tag{9.17}$$

The elements of $\hat{\mathbf{S}}_{i,:}^v$ are rearranged in decreasing order. We obtain $\mathbf{S}_{i,:}^{\prime v} = [S_{i,1}^{\prime v}, \dots, S_{i,n}^{\prime v}]^T$. By considering the constraint $\mathbf{S}_{i,:}^{vT} \mathbf{1} = 1$, and the assumption of using K nearest neighbors to represent each data (i.e., the $\mathbf{S}_{i,:}^v$ vector contains only K non-zero elements), we can get $\alpha_i = \frac{1}{K} - \frac{1}{K} \sum_{l=1}^K S_{i,l}^{\prime v}$. The closed-form solution for $\mathbf{S}_{i,:}^v$ is given by:

$$S_{i,j}^v = S_{i,m}^{\prime v} + \frac{1}{K} - \frac{1}{K} \sum_{l=1}^K S_{i,l}^{\prime v} \quad \text{if } j \in \mathcal{N}_k(i); \quad \text{otherwise } S_{i,j}^v = 0, \tag{9.18}$$

where $m \in \{1, 2, \dots, K\}$ is the corresponding index of the j -th element in the rearranged vector $\mathbf{S}_{i,:}^{\prime v}$.

Update \mathbf{S}^* : Fixing \mathbf{S}^v , \mathbf{P}^v , \mathbf{H} , w_v and δ_v , we have to solve the following problem.

$$\min_{\mathbf{S}^*} \sum_{v=1}^V \|\mathbf{S}^* - \mathbf{S}^v\|_F^2. \quad (9.19)$$

It is clear that the consensus similarity matrix \mathbf{S}^* is equal to the average of all similarity matrices \mathbf{S}^v . Same as before, after obtaining the matrix \mathbf{S}^* , the K-NN algorithm is applied on each row of the matrix \mathbf{S}^* , to get the K most similar samples to each given data point.

Update w_v and δ_v :

After updating the matrices \mathbf{S}^v , \mathbf{S}^* , \mathbf{P}^v , and \mathbf{H} , the weights w_v and δ_v of each view are updated respectively by using Eqs. (9.5) and (9.6). The proposed OSMGSCA method is summarized in **Algorithm 1**.

Algorithm 1 OSMGSCA	
Input:	Data matrices $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$, and their corresponding kernel matrices \mathbf{K}^v . Parameters λ_1 , λ_2 , and λ_3 .
Output:	The consensus nonnegative embedding matrix \mathbf{H} . The similarity graph of each view \mathbf{S}^v . The consensus similarity matrix \mathbf{S}^* . The spectral projection matrices \mathbf{P}^v .
Initialization:	The weights of all views: $w_v = \frac{1}{V}$ and $\delta_v = 1$. Initialize the similarity matrices of all views \mathbf{S}^v as mentioned in section 3.2 Initialize their corresponding spectral projection matrices \mathbf{P}^v as mentioned in section 3.2.
	Repeat
	Update \mathbf{H} using Eq. (9.9).
	Update \mathbf{P}^v , $v = 1, \dots, V$ using Eq. (9.11).
	Update \mathbf{S}^v , $v = 1, \dots, V$ using Eq. (9.18).
	Update \mathbf{S}^* by doing the average of all matrices \mathbf{S}^v .
	Update δ_v , $v = 1, \dots, V$ and w_v , $v = 1, \dots, V$ using Eqs. (9.6) and (9.5) respectively.
	End

9.2.2 Optimization of U-MCJGLNMA (Eq. (9.8))

In this section, the optimization scheme of the objective function in (9.8) is introduced. According to the second variant of our algorithm U-MCJGLNMA, the main difference between this version and OSMGSCA is in computing the matrices \mathbf{S}^v , \mathbf{S}^* , and \mathbf{P}^* . In addition, in this approach there are no weights to update due to the use of consensus matrices in the last two terms. First, we initialize the matrices \mathbf{S}^v and \mathbf{P}^v by using the same method as in [93]. Then, the matrices \mathbf{S}^* and \mathbf{P}^* are initialized by taking the average of the matrices \mathbf{S}^v and \mathbf{P}^v respectively. Therefore, by using an iterative procedure, the algorithm performs the updating steps as follows.

Update \mathbf{H} : Fixing \mathbf{S}^* , \mathbf{S}^v , and \mathbf{P}^* , the nonnegative embedding matrix \mathbf{H} is updated by computing

the derivative of the functional in (9.8) with respect to \mathbf{H} :

$$\frac{\partial f}{\partial \mathbf{H}} = 2\lambda_2 (\mathbf{H} - \mathbf{S}^* \mathbf{P}^*) + 2\lambda_3 \mathbf{L}^* \mathbf{H}.$$

To get the optimal solution \mathbf{H} , we vanish this derivative. Therefore, \mathbf{H} will be given by:

$$\mathbf{H} = (\lambda_2 \mathbf{I} + \lambda_3 \mathbf{L}^*)^{-1} (\lambda_2 \mathbf{S}^* \mathbf{P}^*). \quad (9.20)$$

Same as the OSMGSCA method, to satisfy the orthogonality constraint imposed on the matrix \mathbf{H} , we carry out an orthogonalization step for the matrix \mathbf{H} obtained by Equation (9.20). In addition, to satisfy the constraint of positivity imposed on the matrix \mathbf{H} , we use the element-wise ReLU (Rectified Linear Unit) operator to the elements of the matrix \mathbf{H} .

Update \mathbf{P}^* : Fixing \mathbf{S}^* , \mathbf{S}^v , and \mathbf{H} , the objective function of our method will be equal to:

$$\min_{\mathbf{P}^*} \|\mathbf{S}^* - \mathbf{H} \mathbf{P}^{*T}\|_2^2. \quad (9.21)$$

In a similar way to our first approach, the solution of Equation (9.21) is given by:

$$\mathbf{P}^* = \mathbf{F} \mathbf{T}^T \quad \text{with} \quad \mathbf{F} \mathbf{\Sigma} \mathbf{T}^T = \text{SVD}(\mathbf{S}^{*T} \mathbf{H}). \quad (9.22)$$

Update \mathbf{S}^v : Fixing \mathbf{S}^* , \mathbf{P}^* , \mathbf{H} , we have to solve the following problem.

$$\begin{aligned} \min_{\mathbf{S}^v} \sum_{v=1}^V (Tr(\mathbf{K}^v - 2\mathbf{K}^v + \mathbf{S}^{*T} \mathbf{K}^v \mathbf{S}^v) + \|\mathbf{S}^v\|_2^2 + \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2) \\ \text{s.t. } 0 \leq \mathbf{S}^v \leq \mathbf{1}, (\mathbf{S}^v)^T \mathbf{1} = \mathbf{1}, \text{diag}(\mathbf{S}^v) = \mathbf{0}. \end{aligned} \quad (9.23)$$

By setting the derivative of this equation to zero, the expression of $\hat{\mathbf{S}}^v$, which represents the similarity matrix for each view without taking into account the constraints, can be obtain by the equation below.

$$\hat{\mathbf{S}}^v = (\mathbf{K}^v + (1 + \lambda_1) \mathbf{I})^{-1} (\mathbf{K}^v + \lambda_1 \mathbf{S}^*) \quad (9.24)$$

Then, to satisfy the constraints over each similarity matrix, the same procedure mentioned for the first algorithm on Eq. (9.18) is applied on each similarity matrix.

Update \mathbf{S}^* :

Fixing \mathbf{S}^v , \mathbf{P}^v , and \mathbf{H} , we have to solve the following problem.

$$\min_{\mathbf{S}^*} \sum_{v=1}^V \lambda_1 \|\mathbf{S}^* - \mathbf{S}^v\|_2^2 + \lambda_2 \|\mathbf{S}^* - \mathbf{H}\mathbf{P}^{*T}\|_2^2 + \lambda_3 \text{Tr}(\mathbf{H}^T \mathbf{L}^* \mathbf{H}). \quad (9.25)$$

By vanishing the derivative of Eq. (9.25) with respect to \mathbf{S}^* , the latter can have the following solution:

$$\text{ReLU} \left\{ \left(\left(\sum_{v=1}^V \lambda_1 + \lambda_2 \right) \mathbf{I} \right)^{-1} \left(\sum_{v=1}^V \lambda_1 \mathbf{S}^v + \lambda_2 \mathbf{H}\mathbf{P}^{*T} - \frac{1}{2} \lambda_3 \mathbf{Q} \right) \right\}, \quad (9.26)$$

where the matrix \mathbf{Q} is the distance matrix defined in Eq. (9.13).

Finally, the cluster labels are obtained by taking the maximum of each row of the matrix \mathbf{H} . The proposed U-MCJGLNMA method is summarized in **Algorithm 2**.

Algorithm 2 U-MCJGLNMA	
Input:	Data matrices $\mathbf{X}^v \in \mathbb{R}^{n \times d^v}$, $v = 1, \dots, V$, and their corresponding kernel matrices \mathbf{K}^v . Parameters λ_1 , λ_2 , and λ_3 .
Output:	The consensus nonnegative embedding matrix \mathbf{H} . The similarity graph for each view \mathbf{S}^v . The consensus similarity matrix \mathbf{S}^* . The spectral projection matrices \mathbf{P}^v . The consensus spectral projection matrix \mathbf{P}^* .
Initialization:	Initialize the similarity matrices of all views \mathbf{S}^v and their corresponding spectral projection matrices \mathbf{P}^v . Initialize \mathbf{S}^* and its corresponding \mathbf{P}^* as mentioned before.
	Repeat
	Update \mathbf{H} using Eq. (9.20).
	Update \mathbf{P}^* using Eq. (9.22).
	Update \mathbf{S}^v $v = 1, \dots, V$ using Eq. (9.2.2).
	Update \mathbf{S}^* using Eq. (9.26).
	End

9.3 Performance Evaluation

9.3.1 Experimental Setup

The superiority of our two approaches is demonstrated by using eight real-world multi-view datasets. These datasets are image datasets, with the exception of the BBCSport dataset which is textual dataset.

Compared methods. Our methods are compared with several related state-of-the-art methods including the Co-training multi-view Spectral Clustering method (CotSC) [34], the Co-regularized multi-view Spectral Clustering method (CorSC) in [35]. Besides, some graph-based methods are used for comparison such as: Multi-view Learning clustering with Adaptive Neighbors approach (MLAN) [44], the Self-weighted Multi-view Clustering with multiple graphs method (SwMC) [95],

the Affinity Aggregation for Spectral Clustering method (AASC) [96], the Graph Learning for Multi-View clustering approach (MVGL) [15], the Parameter-free Auto-weighted Multiple Graph Learning method (AMGL) [13], the Multi-view clustering via Adaptively Weighted Procrustes method (AWP) [20]. Other recent methods are used for the comparison and are adopted for some datasets: the Auto-weighted Multi-View Clustering via Kernelized graph learning method (MVCSK) described on [24], the Multi-view spectral clustering via integrating Nonnegative Embedding and Spectral Embedding method (NESE) [11], the Sparse Multi-View Spectral Clustering via graph learning method (S-MVSC) [77], the Consistency-aware and Inconsistency-aware Graph-based Multi-View Clustering (CI-GMVC) introduced in [78], the novel approach called Multi-View Clustering in Latent Embedding Space (MCLES) in [67], and the multi-view spectral clustering via Constrained Nonnegative Embedding (CNESE) presented in [97]. In addition, Spectral Clustering Best (SC-Best) [12], which implements the spectral clustering algorithm on each view separately, and then report the best result obtained for the best view, is also used as a competitive method.

In our two approaches, the initialization of the matrices \mathbf{S}^v and \mathbf{P}^v follows the same computational scheme as in [11], and the matrices \mathbf{S}^* and \mathbf{P}^* are initialized as mentioned before.

Parameter settings. According to our algorithm, there are three main parameters to be determined: λ_1 , λ_2 and λ_3 . The value of λ_1 is chosen in the set $\{0.0005, 0.005, 0.05, 0.03, 1, 10, \text{ and } 10^{+2}\}$, the value of λ_2 is chosen in the set $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$, and the value of λ_3 is chosen in the set $\{0.005, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 10, 10^{+2}, 10^{+3}\}$. Besides, we use a Gaussian kernel. Moreover, the two proposed methods needs to set the K nearest samples of a given data point. Its value indicates the number of non-zero elements in each row of the similarity matrices for all views, and the consensus similarity matrix. This value is chosen from 10 to 20. The detailed study of this parameter sensitivity is presented in Section 4.3.

There are four well-known evaluation metrics used in this chapter: Clustering Accuracy (ACC), Normalized Mutual Information (NMI), Purity, and Adjusted Rand Index (ARI) [106, 107]. For the above indicators, the higher value means the result is better.

9.3.2 Experimental results

The comparison results in terms of ACC, NMI, Purity, and ARI obtained by the proposed methods OSMGSCA and U-MCJGLNMA and all competing methods on the eight datasets, are reported in Tables 9.1 and 9.2. In these tables, the highest values (best performances) are marked in bold,

and the standard deviations are presented in parentheses to indicate the variation of the value over several trials. Note that the methods that provide a direct clustering solution this standard deviation was set to zero. According to Table 9.1, which reports the experimental results of the ORL, COIL20 and Out-Scene datasets, the best results are obtained by using the recent methods which are: MVSCK, NESE, S-MVSC, CI-GMVC, MCLES, CNESE, and our proposed methods: OSMGSCA and U-MCJGLNMA. Then, these methods are adopted for comparison for the rest of the datasets. Table 9.2 shows a comparison between our two methods and the mentioned state-of-the-art methods on the BBCSport, MSRCv1, UCI Digits, Caltech101-7, and MNIST-10000 datasets. From these two tables, we can see the two proposed methods OSMGSCA and U-MCJGLNMA outperformed the other competing methods in all eight datasets. Furthermore, we found that the U-MCJGLNMA method was better than the OSMGSCA method for all datasets with the exception of the Caltech101-7 dataset.

9.3.3 Ablation study

The proposed criterion (9.8) consists of three main terms: the consensus graph construction, the convolution term, and the cluster label smoothness term. We create three different variants of our method with different configurations to demonstrate the importance of the suggested criterion and its components. These three distinct models are: UMCJGLNMA-SC, UMCJGLNMA-C, and UMCJGLNMA-S. (1) No convolution and smoothness term in the objective function (9.8) (i.e., λ_2 and λ_3 are set to zero), and we call the obtained method UMCJGLNMA-SC because it is simplified to a coherent graph construction accompanied by a spectral clustering stage, (2) No convolution term (λ_2 is set to zero), and the obtained approach is called UMCJGLNMA-S, and (3) No smoothness term (λ_3 is set to zero), and the obtained approach is called UMCJGLNMA-C. The results obtained by adopting UMCJGLNMA-SC, UMCJGLNMA-S, and UMCJGLNMA-C are presented in Table 9.3. Two datasets are used: ORL and MSRCv1.

From Table 9.3, the results obtained with UMCJGLNMA-SC indicate that the use of the consensus similarity matrix without the last two terms can lead to bad performance, which indicates the importance of the last two terms in our objective function. For the ORL dataset, it can be seen from the table that the convolution term has a significant impact on the clustering results. However, for the MSRCv1 dataset, the smoothness term is more relevant than the convolution term. This is normal and it is related to the various datasets employed in this study.

All of these results indicated that including all terms in the objective function led to the high clustering efficiency of our suggested method.

Table 9.1: Clustering performance on the COIL20, ORL, and Out-Scene datasets.

Dataset	Method	ACC	NMI	Purity	ARI
ORL	SC-Best [33]	0.66 (± 0.02)	0.76 (± 0.02)	0.71 (± 0.02)	0.67 (± 0.01)
	AWP [20]	0.80 (± 0.00)	0.91 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MLAN [44]	0.78 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.67 (± 0.00)
	SwMC [95]	0.77 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)	0.62 (± 0.00)
	AMGL [13]	0.75 (± 0.02)	0.90 (± 0.02)	0.82 (± 0.02)	0.63 (± 0.09)
	AASC [96]	0.82 (± 0.02)	0.91 (± 0.01)	0.85 (± 0.01)	0.76 (± 0.02)
	MVGL [15]	0.75 (± 0.00)	0.88 (± 0.00)	0.80 (± 0.00)	0.55 (± 0.00)
	CorSC [35]	0.77 (± 0.03)	0.90 (± 0.01)	0.82 (± 0.03)	0.72 (± 0.04)
	CotSC [34]	0.75 (± 0.04)	0.87 (± 0.01)	0.78 (± 0.03)	0.67 (± 0.03)
	MVCSK [24]	0.85 (± 0.02)	0.94 (± 0.01)	0.88 (± 0.02)	0.81 (± 0.02)
	NESE [11]	0.82 (± 0.00)	0.91 (± 0.00)	0.85 (± 0.00)	0.75 (± 0.00)
	S-MVSC [77]	0.80 (± 0.02)	0.93 (± 0.01)	0.82 (± 0.02)	0.89 (± 0.01)
	CI-GMVC [78]	0.81 (± 0.00)	0.92 (± 0.00)	0.85 (± 0.00)	0.74 (± 0.00)
	MCLES [67]	0.84 (± 0.00)	0.94 (± 0.00)	0.88 (± 0.00)	0.79 (± 0.00)
	CNESE [97]	0.87 (± 0.00)	0.95 (± 0.00)	0.89 (± 0.00)	0.84 (± 0.00)
	OSMGSCA	0.90 (± 0.00)	0.96 (± 0.00)	0.93 (± 0.00)	0.88 (± 0.00)
	U-MCJGLNMA	0.93 (± 0.00)	0.97 (± 0.00)	0.93 (± 0.00)	0.90 (± 0.00)
COIL20	SC-Best [33]	0.73 (± 0.01)	0.82 (± 0.01)	0.75 (± 0.01)	0.68 (± 0.02)
	AWP [20]	0.68 (± 0.00)	0.87 (± 0.00)	0.75 (± 0.00)	0.71 (± 0.00)
	MLAN [44]	0.84 (± 0.00)	0.92 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)
	SwMC [95]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.84 (± 0.00)
	AMGL [13]	0.80 (± 0.04)	0.91 (± 0.02)	0.85 (± 0.03)	0.74 (± 0.07)
	AASC [96]	0.79 (± 0.00)	0.89 (± 0.00)	0.83 (± 0.00)	0.76 (± 0.00)
	MVGL [15]	0.78 (± 0.00)	0.88 (± 0.00)	0.81 (± 0.00)	0.75 (± 0.00)
	CorSC [35]	0.68 (± 0.04)	0.78 (± 0.02)	0.70 (± 0.03)	0.62 (± 0.03)
	CotSC [34]	0.70 (± 0.03)	0.80 (± 0.02)	0.72 (± 0.03)	0.65 (± 0.03)
	MVCSK [24]	0.65 (± 0.04)	0.80 (± 0.02)	0.70 (± 0.03)	0.61 (± 0.05)
	NESE [11]	0.77 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.69 (± 0.00)
	S-MVSC [77]	0.62 (± 0.01)	0.86 (± 0.02)	0.77 (± 0.02)	0.97 (± 0.02)
	CI-GMVC [78]	0.86 (± 0.00)	0.94 (± 0.00)	0.90 (± 0.00)	0.83 (± 0.00)
	MCLES [67]	0.79 (± 0.00)	0.88 (± 0.00)	0.83 (± 0.00)	0.75 (± 0.00)
	CNESE [97]	0.82 (± 0.00)	0.88 (± 0.00)	0.82 (± 0.00)	0.78 (± 0.00)
	OSMGSCA	0.92 (± 0.00)	0.97 (± 0.00)	0.95 (± 0.00)	0.93 (± 0.00)
	U-MCJGLNMA	1.00 (± 0.00)	1.00 (± 0.00)	1.00 (± 0.00)	1.00 (± 0.00)
Out-Scene	SC-best [33]	0.47 (± 0.01)	0.39 (± 0.01)	0.57 (± 0.01)	0.34 (± 0.01)
	AWP [20]	0.65 (± 0.00)	0.51 (± 0.00)	0.65 (± 0.00)	0.42 (± 0.00)
	MLAN [44]	0.55 (± 0.02)	0.47 (± 0.01)	0.55 (± 0.02)	0.33 (± 0.03)
	SwMC [95]	0.50 (± 0.00)	0.47 (± 0.00)	0.50 (± 0.00)	0.38 (± 0.00)
	AMGL [13]	0.51 (± 0.05)	0.45 (± 0.03)	0.52 (± 0.04)	0.34 (± 0.05)
	AASC [96]	0.60 (± 0.00)	0.48 (± 0.00)	0.60 (± 0.00)	0.35 (± 0.00)
	MVGL [15]	0.42 (± 0.00)	0.31 (± 0.00)	0.43 (± 0.00)	0.16 (± 0.00)
	CorSC [35]	0.51 (± 0.04)	0.39 (± 0.03)	0.52 (± 0.03)	0.31 (± 0.02)
	CotSC [34]	0.38 (± 0.02)	0.22 (± 0.01)	0.39 (± 0.02)	0.16 (± 0.01)
	MVCSK [24]	0.65 (± 0.01)	0.52 (± 0.00)	0.65 (± 0.01)	0.42 (± 0.00)
	NESE [11]	0.63 (± 0.00)	0.53 (± 0.00)	0.66 (± 0.00)	0.46 (± 0.00)
	S-MVSC [77]	0.48 (± 0.01)	0.54 (± 0.02)	0.65 (± 0.01)	0.46 (± 0.04)
	CI-GMVC [78]	0.35 (± 0.01)	0.31 (± 0.00)	0.35 (± 0.01)	0.19 (± 0.00)
	MCLES [67]	0.65 (± 0.00)	0.53 (± 0.00)	0.67 (± 0.00)	0.46 (± 0.00)
	CNESE [97]	0.66 (± 0.00)	0.55 (± 0.00)	0.67 (± 0.00)	0.47 (± 0.00)
	OSMGSCA	0.69 (± 0.00)	0.54 (± 0.00)	0.69 (± 0.00)	0.45 (± 0.00)
	U-MCJGLNMA	0.70 (± 0.00)	0.59 (± 0.00)	0.70 (± 0.00)	0.50 (± 0.00)

9.3.4 Parameter sensitivity

According to **Algorithm 1**, there are three explicit parameters to be analysed: λ_1 , λ_2 , and λ_3 . In addition the parameter K , which is used through the iterations of our algorithms, should be tuned. Besides, according to T_0 , which represents the parameter used to control the scaling of the Gaussian kernel, the experiments show that its best value is equal to 2. Then, we fixed T_0 to 2. We analysed the sensitivity of these parameters on the two datasets ORL and MSRCv1 for our two methods OSMGSCA and U-MCJGLNMA.

Figure 9.3 shows the clustering performance indicators ACC and NMI, obtained by the OSMGSCA method, as a function of the parameters λ_1 and λ_2 , for the ORL dataset (upper row) and the

Table 9.2: Clustering performance on the BBCSport, MSRCv1, Handwritten, Extended-Yale and MNIST-10000 datasets.

Dataset	Method	ACC	NMI	Purity	ARI
BBCSport	MVCSK	0.90 (± 0.07)	0.82 (± 0.02)	0.90 (± 0.02)	0.85 (± 0.07)
	NESE	0.72 (± 0.00)	0.69 (± 0.00)	0.75 (± 0.00)	0.60 (± 0.00)
	S-MVSC	0.58 (± 0.07)	0.67 (± 0.01)	0.73 (± 0.02)	0.83 (± 0.04)
	CI-GMVC	0.61 (± 0.00)	0.46 (± 0.00)	0.63 (± 0.00)	0.36 (± 0.00)
	MCLES	0.88 (± 0.00)	0.80 (± 0.00)	0.88 (± 0.00)	0.83 (± 0.00)
	CNESE	0.72 (± 0.00)	0.68 (± 0.00)	0.76 (± 0.00)	0.60 (± 0.00)
	OSMGSCA	0.90 (± 0.00)	0.84 (± 0.00)	0.90 (± 0.00)	0.88 (± 0.00)
	U-MCJGLNMA	0.98 (± 0.00)	0.94 (± 0.00)	0.98 (± 0.00)	0.95 (± 0.00)
MSRCv1	MVCSK	0.70 (± 0.02)	0.59 (± 0.03)	0.70 (± 0.02)	0.50 (± 0.04)
	NESE	0.77 (± 0.00)	0.72 (± 0.00)	0.80 (± 0.00)	0.64 (± 0.00)
	S-MVSC	0.60 (± 0.00)	0.69 (± 0.02)	0.74 (± 0.02)	0.79 (± 0.01)
	CI-GMVC	0.74 (± 0.00)	0.72 (± 0.00)	0.77 (± 0.00)	0.59 (± 0.00)
	MCLES	0.90 (± 0.01)	0.83 (± 0.02)	0.90 (± 0.01)	0.77 (± 0.00)
	CNESE	0.86 (± 0.00)	0.76 (± 0.00)	0.86 (± 0.00)	0.72 (± 0.00)
	OSMGSCA	0.92 (± 0.00)	0.85 (± 0.00)	0.92 (± 0.00)	0.82 (± 0.00)
	U-MCJGLNMA	0.96 (± 0.00)	0.91 (± 0.00)	0.96 (± 0.00)	0.90 (± 0.00)
UCI Digits	MVCSK	0.81 (± 0.00)	0.79 (± 0.01)	0.81 (± 0.00)	0.71 (± 0.01)
	NESE	0.78 (± 0.00)	0.83 (± 0.00)	0.78 (± 0.00)	0.75 (± 0.00)
	S-MVSC	0.71 (± 0.00)	0.79 (± 0.01)	0.75 (± 0.00)	0.77 (± 0.01)
	CI-GMVC	0.88 (± 0.00)	0.89 (± 0.00)	0.88 (± 0.00)	0.85 (± 0.00)
	MCLES	0.82 (± 0.01)	0.83 (± 0.05)	0.85 (± 0.00)	0.80 (± 0.01)
	CNESE	0.79 (± 0.00)	0.83 (± 0.00)	0.79 (± 0.00)	0.76 (± 0.00)
	OSMGSCA	0.88 (± 0.00)	0.89 (± 0.00)	0.88 (± 0.00)	0.85 (± 0.00)
	U-MCJGLNMA	0.89 (± 0.00)	0.90 (± 0.00)	0.89 (± 0.00)	0.85 (± 0.00)
Caltech101-7	MVCSK	0.57 (± 0.02)	0.51 (± 0.02)	0.83 (± 0.01)	0.45 (± 0.03)
	NESE	0.67 (± 0.00)	0.55 (± 0.00)	0.87 (± 0.00)	0.52 (± 0.00)
	S-MVSC	0.64 (± 0.03)	0.55 (± 0.02)	0.72 (± 0.01)	0.51 (± 0.03)
	CI-GMVC	0.74 (± 0.00)	0.54 (± 0.00)	0.85 (± 0.00)	0.48 (± 0.00)
	MCLES	0.74 (± 0.00)	0.64 (± 0.00)	0.92 (± 0.00)	0.62 (± 0.00)
	CNESE	0.69 (± 0.00)	0.58 (± 0.00)	0.88 (± 0.00)	0.56 (± 0.00)
	OSMGSCA	0.86 (± 0.00)	0.70 (± 0.00)	0.90 (± 0.00)	0.74 (± 0.00)
	U-MCJGLNMA	0.75 (± 0.00)	0.65 (± 0.00)	0.93 (± 0.00)	0.63 (± 0.00)
MNIST-10000	MVCSK	0.49 (± 0.00)	0.41 (± 0.00)	0.50 (± 0.00)	0.29 (± 0.00)
	NESE	0.81 (± 0.00)	0.83 (± 0.00)	0.85 (± 0.00)	0.76 (± 0.00)
	S-MVSC	0.77 (± 0.01)	0.81 (± 0.01)	0.81 (± 0.02)	0.76 (± 0.07)
	CI-GMVC	0.66 (± 0.00)	0.71 (± 0.00)	0.71 (± 0.00)	0.51 (± 0.00)
	MCLES	0.80 (± 0.00)	0.83 (± 0.00)	0.85 (± 0.00)	0.77 (± 0.00)
	CNESE	0.81 (± 0.00)	0.83 (± 0.00)	0.86 (± 0.00)	0.78 (± 0.00)
	OSMGSCA	0.82 (± 0.00)	0.86 (± 0.00)	0.86 (± 0.00)	0.79 (± 0.00)
	U-MCJGLNMA	0.82 (± 0.00)	0.87 (± 0.00)	0.87 (± 0.00)	0.81 (± 0.00)

Table 9.3: Ablation study with different models. The best performance for each indicator is in bold.

Dataset	Variant	Consensus graph construction	Convolution term	Smoothness term	ACC	NMI	Purity	ARI
ORL	UMCJGLNMA-SC	✓	✗	✗	0.80	0.85	0.71	0.77
	UMCJGLNMA-S	✗	✗	✓	0.83	0.91	0.85	0.77
	UMCJGLNMA-C	✗	✓	✗	0.89	0.94	0.89	0.84
	U-MCJGLNMA	✓	✓	✓	0.93	0.97	0.93	0.90
MSRCv1	UMCJGLNMA-SC	✓	✗	✗	0.70	0.59	0.70	0.51
	UMCJGLNMA-S	✗	✗	✓	0.75	0.63	0.72	0.55
	UMCJGLNMA-C	✗	✓	✗	0.62	0.57	0.69	0.50
	U-MCJGLNMA	✓	✓	✓	0.96	0.91	0.96	0.90

MSRCv1 dataset (lower row). The four sub-figures (a), (b), (c) and (d) of this figure depict the variation of the indicators as a function of parameters λ_1 and λ_2 when the parameters λ_3 and K are fixed. For the ORL dataset, according to Figure 9.3, the best ACC and NMI are obtained when λ_1 and λ_2 are equal to 0.05 and 0.5. Concerning the MSRCv1 dataset, it is clear from Figure 9.3 that the best values of ACC and NMI are obtained for a value of λ_1 and λ_2 equal to 10 and 0.3 for

the OSMGSCA method.

Besides, the same parameter study is repeated for the U-MCJGLNMA method. Figure 9.4 shows the clustering performance indicators ACC and NMI, obtained by the U-MCJGLNMA method, as a function of the parameters λ_1 and λ_2 , for the ORL dataset and the MSRCv1 dataset. For the ORL dataset, according to Figure 9.4, the best ACC and NMI are obtained when λ_1 and λ_2 are equal to 1 and 0.5. Concerning the MSRCv1 dataset, it is clear from Figure 9.4 that the best values of ACC and NMI are obtained for a value of λ_1 and λ_2 equal to 1 and 10^{-4} for the U-MCJGLNMA method. In addition, Figure 9.5 analyzed the effect of parameter λ_3 on clustering results, whereas the other three parameters are set to their optimal values for the ORL and MSRCv1 datasets. According to these sub-figures, the best values of ACC and NMI for the OSMGSCA method are obtained for a parameter λ_3 equal to 0.005 for the ORL dataset, and equal to 0.03 for the MSRCv1 dataset.

Also, concerning the U-MCJGLNMA method, it is clear from Figure 9.5 that the best values of ACC and NMI are obtained for a parameter λ_3 equal to 0.05 for the ORL dataset, and equal to 0.5 for the MSRCv1 dataset.

The effect of the parameter K on the clustering results is analysed in Figure 9.6, while the other three parameters are fixed to their best values for each dataset. According to this figure, the best values of ACC and NMI, for the OSMGSCA method used on the ORL dataset, are obtained for a K equal to 18 and 19, respectively. For the MSRCv1 dataset, these values for K are equal to 16. Concerning the U-MCJGLNMA method, the best values of ACC and NMI are obtained for a parameter K equal to 11 for the ORL dataset, and equal to 16 and 17 for the MSRCv1 dataset.

9.3.5 Analysis of results and method comparison

Several competing methods are used to test the effectiveness of our proposed methods. First, the result of "SC -Best", which is the best result achieved by applying the spectral clustering method to each view, is, in most cases, worse than the results of most multi-view clustering techniques.

From the overall results presented on Tables 9.1 and 9.2, our two proposed methods outperform all other competing methods on all datasets. Besides, the U-MCJGLNMA method presents better performance than the OSMGSCA method except for the Caltech101-7 dataset presented in Table 9.2. However, it achieves better performance than that of all other competing methods. Furthermore, Table 9.2 (lower part) reports the result of our two proposed methods on the MNIST-10000 dataset. These two methods outperform all other methods, showing that they perform well on large datasets. All these results indicate that using unified matrices and enforcing the single view

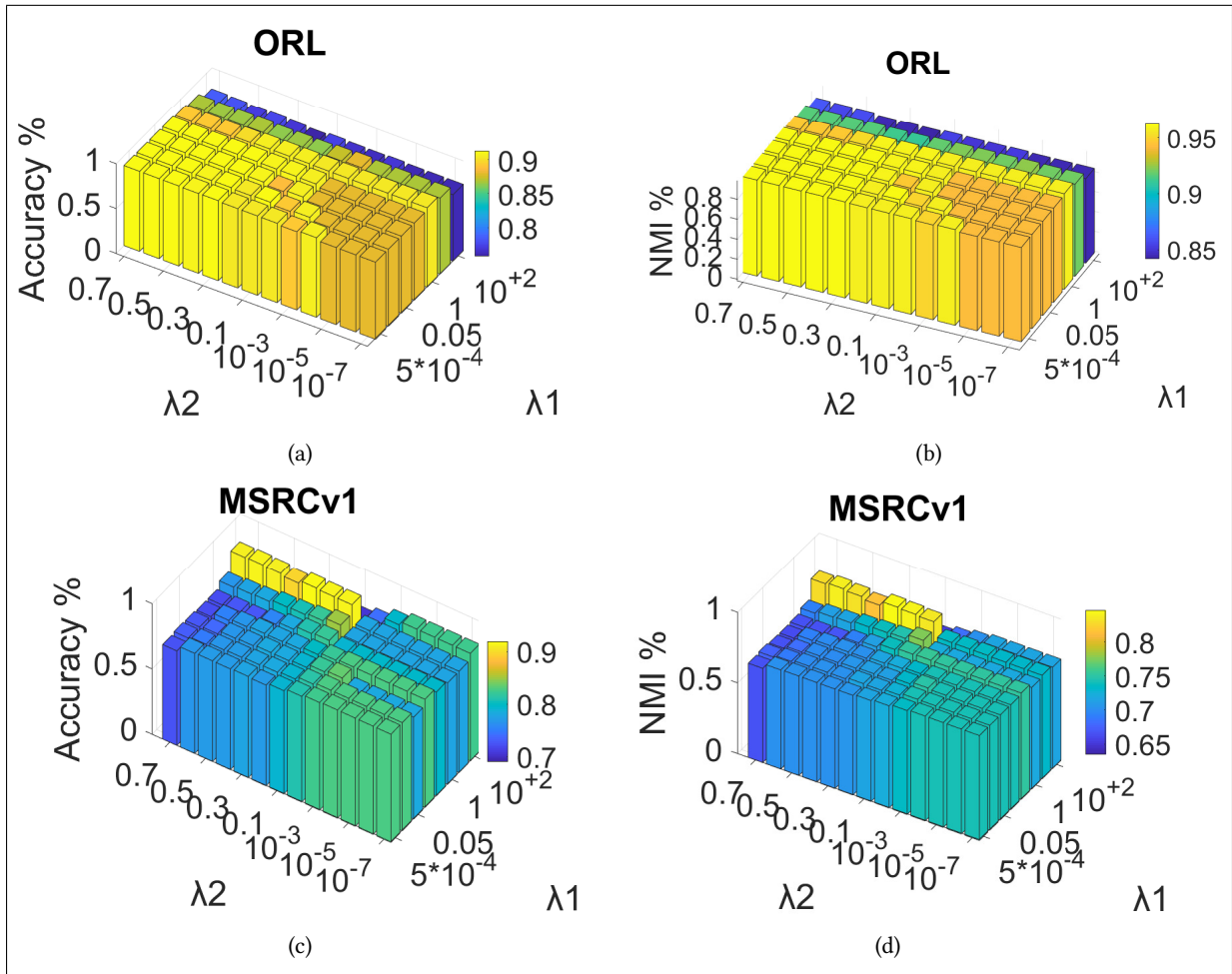


Figure 9.3: Clustering performance ACC (%) and NMI (%) of the OSMGSCA method as a function of λ_1 , λ_2 on the ORL and MSRCv1 datasets.

matrices to be similar to each other can, without any doubt, enhance the clustering results.

9.3.6 Computational complexity

In this section, the computational complexity of the proposed OSMGSCA method is studied. Our algorithm consists of six main steps: updating \mathbf{H} , \mathbf{P}^v , \mathbf{S}^v , \mathbf{S}^* , δ_v and w_v (see **Algorithm 1**). The calculation of the V kernel matrices has a computational cost of $\mathcal{O}(n^2k)$ where k is the sum of the dimensions of the instances in the V views. Thus, $k = d^1 + d^2 + \dots + d^V$.

By inspecting the six steps of **Algorithm 1**, we can see easily that steps 1, 2 and 3 are the most expensive ones. Indeed, step 4 consists of matrix addition and division (Doing the average of the similarity matrices of all views). Moreover, steps 5 and 6 contain simple matrix multiplications and additions. Therefore, we can ignore their computation cost.

To get the matrix \mathbf{H} (step 1), a matrix inversion of an $n \times n$ matrix is needed (or equivalently we should solve a linear system whose square matrix size is $n \times n$). If the orthogonalization of

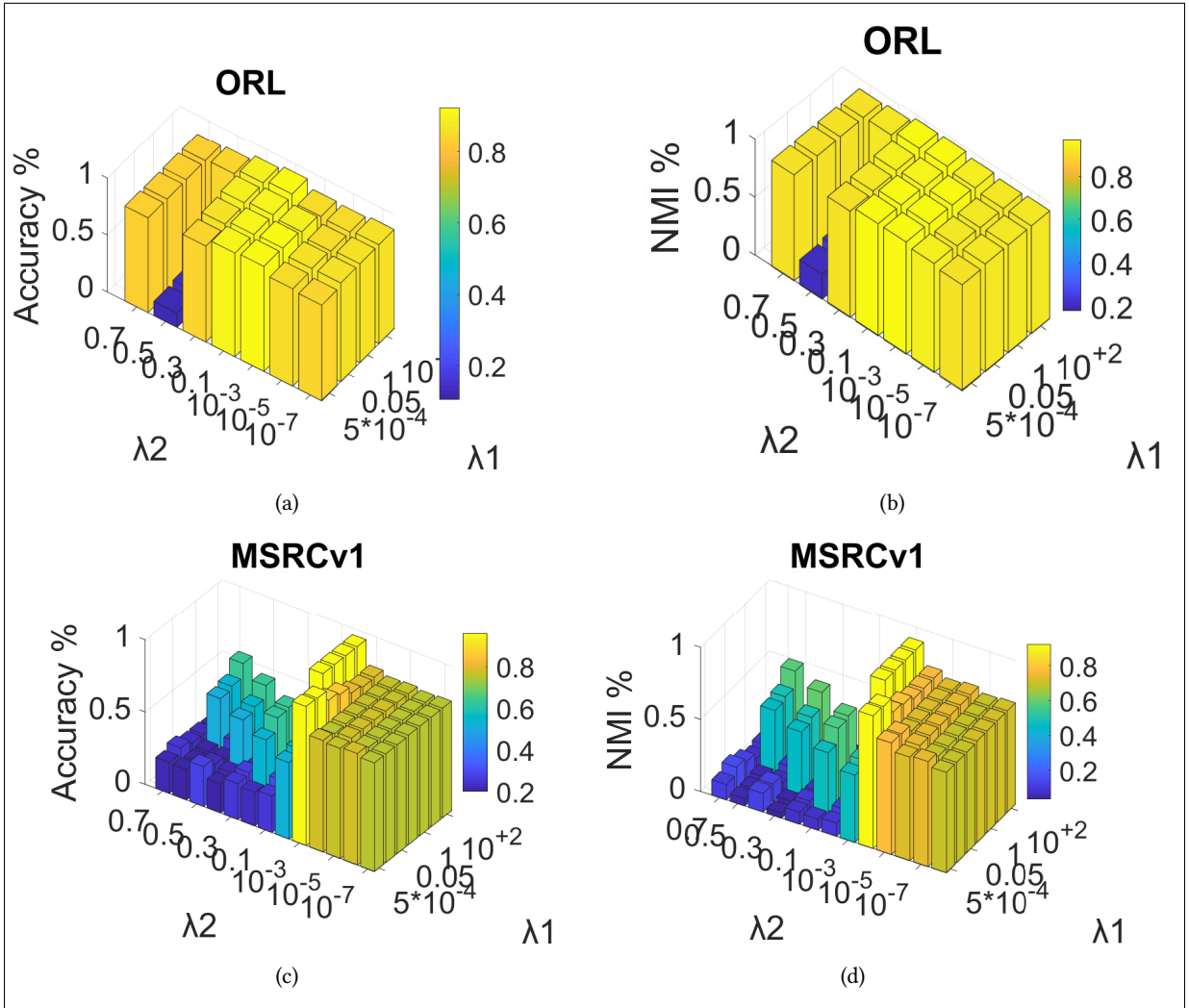


Figure 9.4: Clustering performance ACC (%) and NMI (%) of the U-MCJGLNMA method as a function of λ_1 , λ_2 on the ORL and MSRCv1 datasets.

the matrix \mathbf{H} is invoked, one should add the associated cost which is $\mathcal{O}(nC^2)$ where C is the number of clusters. To estimate the matrix \mathbf{P}^v (step 2), we should compute the SVD of the matrix $(\mathbf{S}^{vT} \mathbf{H})$. The computational cost of this is $\mathcal{O}(nC^2)$, where C is the total number of clusters. To estimate the graph matrix of each view \mathbf{S}^v (step 3), one matrix inversion of size $n \times n$ is needed. Thus, the computational cost of the third step is $\mathcal{O}(n^3)$.

Let τ be the number of iterations of the proposed iterative algorithm. The overall computational complexity of the proposed method will be $\mathcal{O}(n^2k + \tau(2nC^2 + n^3))$.

9.3.7 Convergence Study

In this section, the convergence of the two proposed methods OSMGSCA and U-MCJGLNMA is investigated. For a case study, we consider the two datasets: ORL and MSRCv1. The maximum

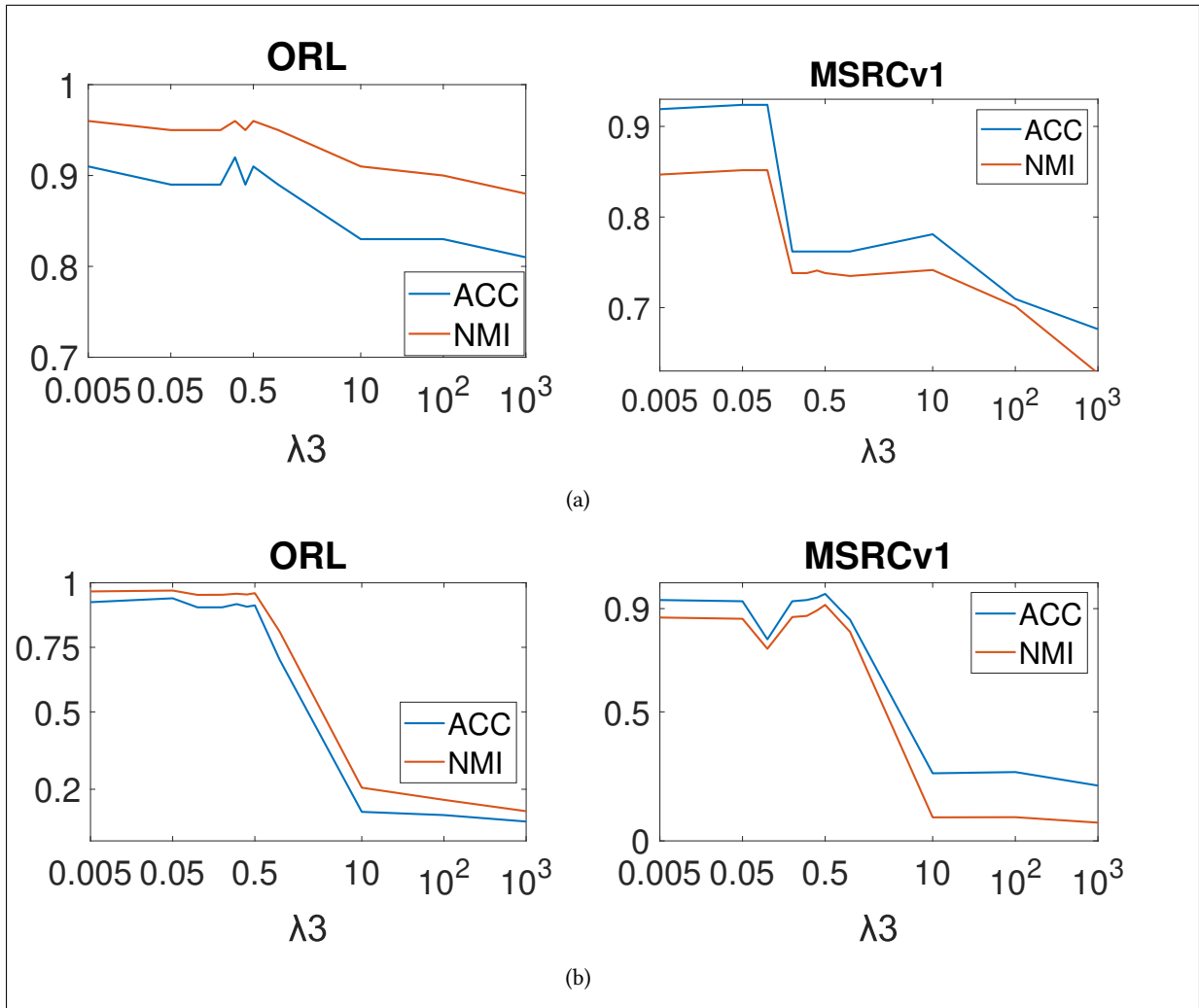


Figure 9.5: Clustering performance ACC (%) and NMI (%) as a function of λ_3 on the ORL and MSRCv1 datasets: (a): OSMGSCA, (b) U-MCJGLNMA.

number of iterations in our two methods is fixed to 100 iterations. Figure 9.7 depicts the value of the objective function given by Eqs. (9.7) and (9.8) for the two methods OSMGSCA and U-MCJGLNMA, respectively. From this Figure, it can be seen that our methods have strong and stable convergence properties. The objective function decreases rapidly with increasing number of iterations. Convergence is reached in less than 30 iterations for the OSMGSCA method and in less than 20 iterations for U-MCJGLNMA method.

9.4 Clustering visualization

In this section, we provide the visualization of the clustering obtained by different methods when applied on the ORL dataset. This is illustrated in Figure 9.8.

This visualization is achieved by using the t-Distributed Stochastic Neighbor Embedding method

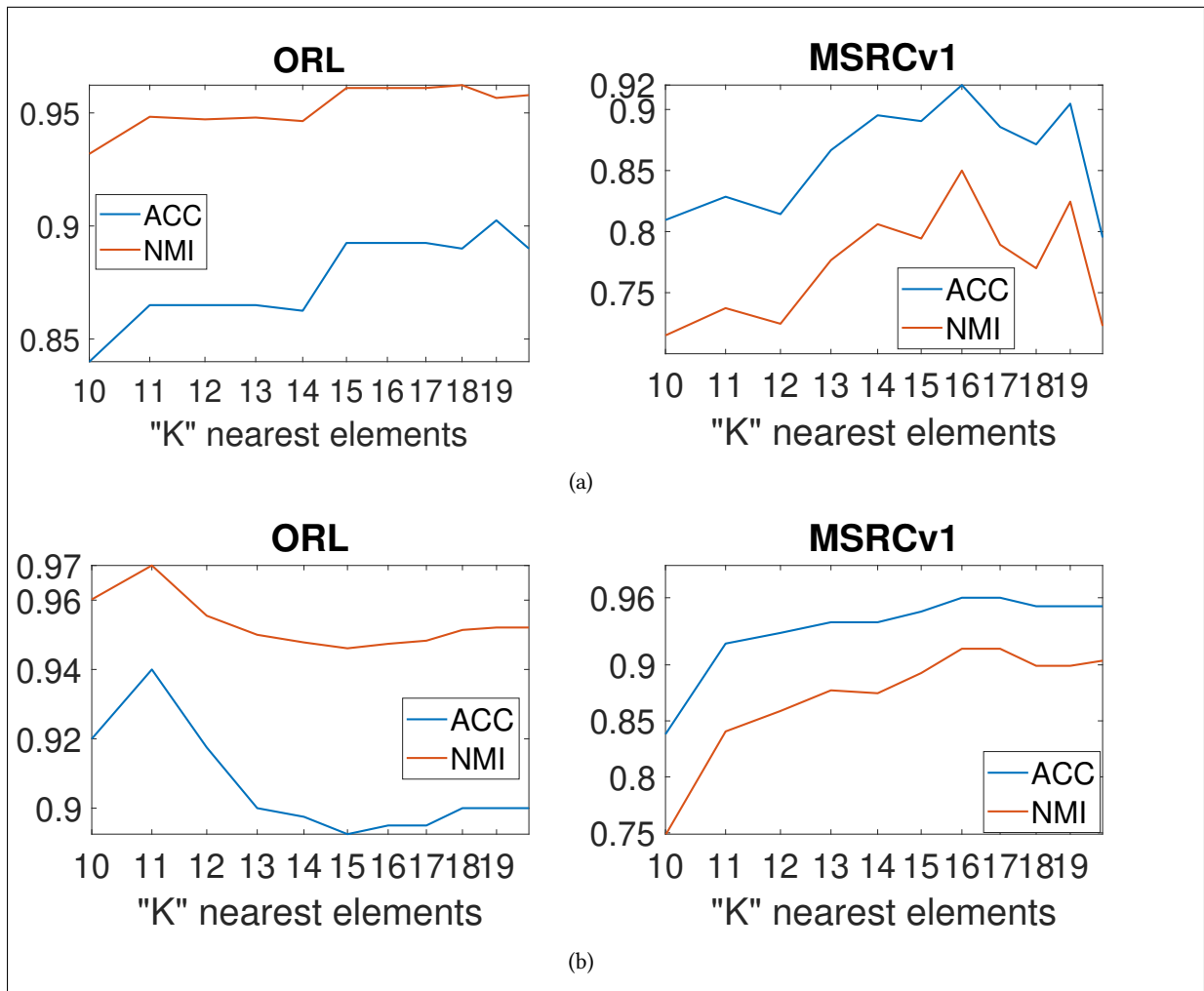


Figure 9.6: Clustering performance ACC (%) and NMI (%) as a function of K on the ORL and MSRCv1 datasets: (a): OSMGSCA, (b): U-MCJGLNMA.

(t-SNE) [98] which maps the different matrices (data representation or soft clustering assignments) obtained by the MVCSK, NESE, S-MVSC, CI-GMVC, MCLES, CNESE, OSMGSCA and U-MCJGLNMA methods for the ORL dataset, into points in a 2D space where they are visualized. The corresponding rows of each matrix are set as input for t-SNE.

The colors in all sub-images correspond to the estimated clusters, and each point represents a sample. The results from Figure 9.8 indicate that our OSMGSCA method gives more separated clusters than other methods, and it is worse than those obtained by our U-MCJGLNMA method. These results are consistent with the experimental results depicted in Table 9.1.

9.5 Conclusion

In this chapter, we proposed two new multi-view clustering approaches. The first one is a novel one-step multi-view clustering algorithm that uses a consensus similarity matrix extracted from

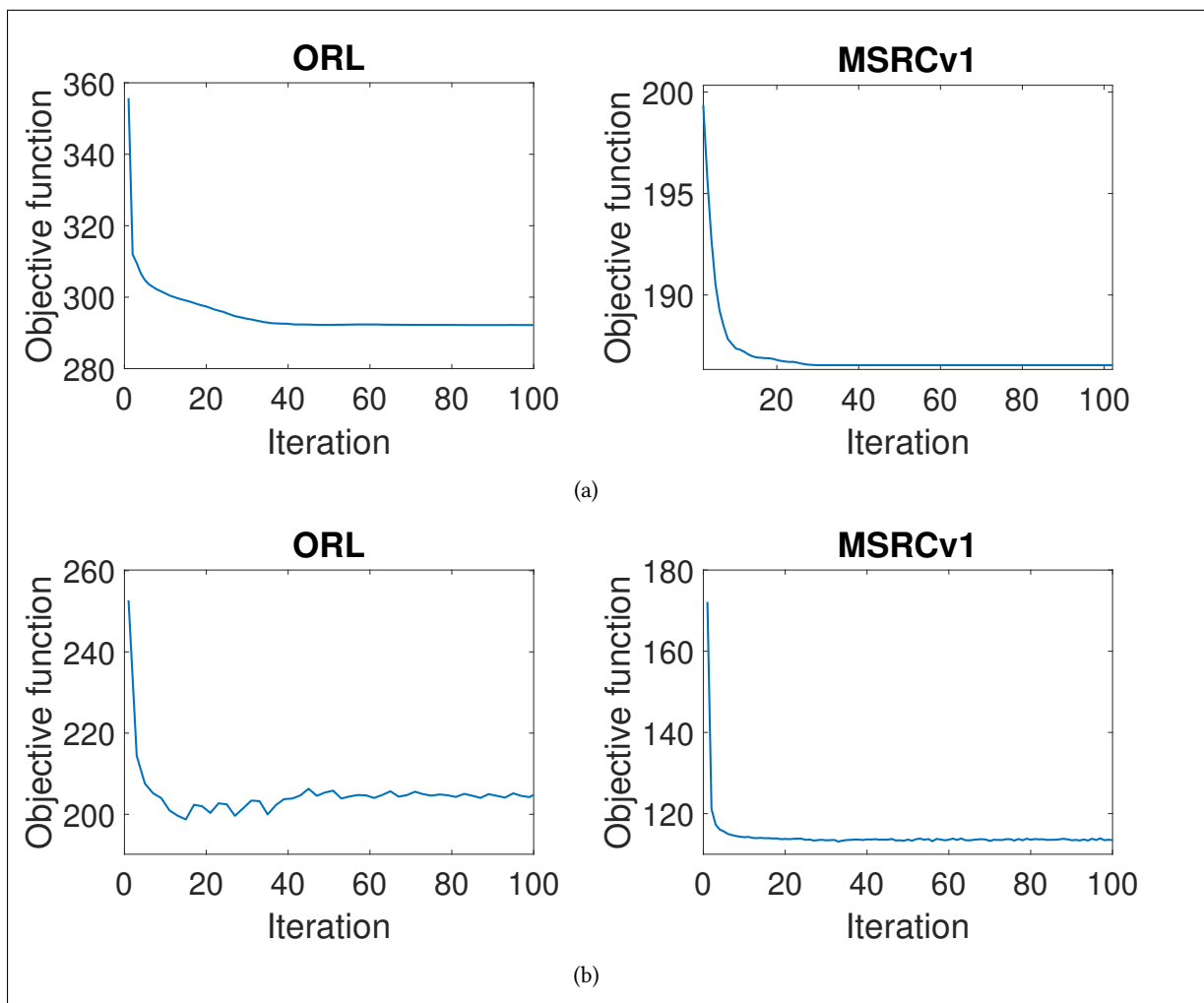


Figure 9.7: Convergence of our two proposed methods: (a) OSMGSCA and (b) U-MCJGLNMA on the ORL and MSRCv1 datasets.

different views and a nonnegative embedding matrix to give the final clustering assignment. Besides, an improvement over the objective function of this new method gives another version of our proposed method, based on unified matrices, and reaches better performance. Different from most existing methods, our methods borrow the idea of multi-view consensus clustering and perform the consensus representation of all views and the clustering assignment into a unified framework. In addition, these two methods use many constraints on the cluster index matrix, such as the cluster label smoothness constraint to improve the overall performance. Extensive experiments carried out on eight multi-view real-world datasets with different sizes and types show that both, OSMGSCA and U-MCJGLNMA, outperform the competing state-of-the-art multi-view clustering methods.

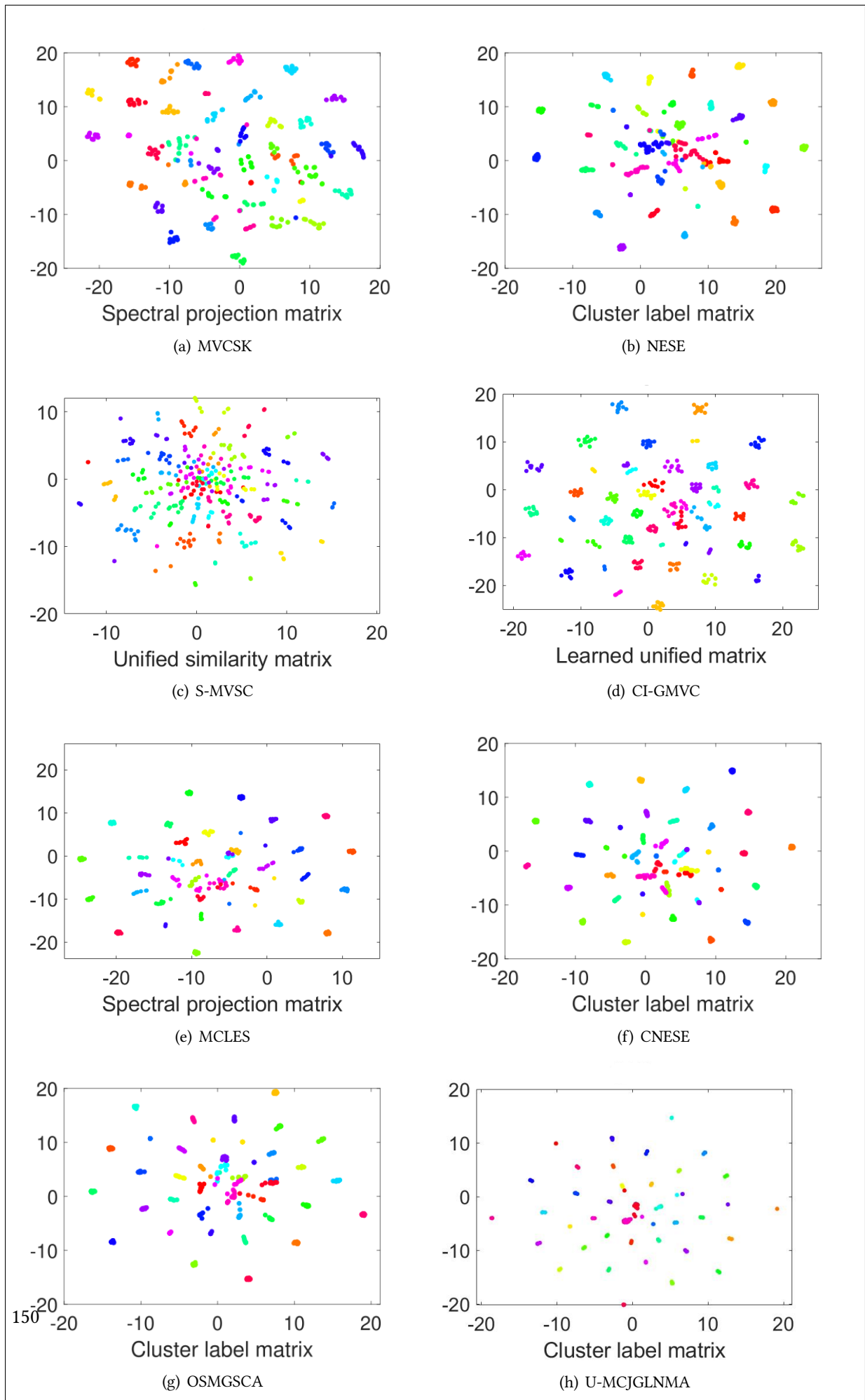


Figure 9.8: t-SNE visualization of the clustering obtained by four different methods on the ORL dataset.

Conclusion and future work

10.1 Conclusion

In this thesis, different multi-view clustering methods are presented. The first method is called Multi-view Clustering via Kernelized Graph and Nonnegative Embedding (MKGNE) and is presented in Chapter 5. Unlike other multi-view clustering methods, this method integrates the ideas of nonnegative matrix factorization methods and graph-based methods. It provides the final cluster assignment directly from the nonnegative embedding matrix, which is used as a soft cluster assignment matrix. Moreover, this matrix is a kind of convolution of the consensus spectral data representation over the graph. Moreover, this method can calculate all unknown variables together, namely: 1) the unified similarity matrix, 2) the unified spectral projection matrix, 3) the nonnegative embedding matrix, and 4) the weight of each view automatically.

A variant of the previous idea is presented in Chapter 6. Like the MKGNE technique, the new variant can overcome some of the shortcomings of existing multi-view clustering methods. Moreover, it can compute the four subtasks mentioned above simultaneously. The method is called Multi-view Clustering via Consensus Graph Learning and Nonnegative Embedding (MVCGE) and differs from the previous method by the third term in the objective function and the orthogonality constraint over the matrix \mathbf{H} .

The method presented in Chapter 7 is called Constrained Nonnegative Embedding and Spectral Embedding (CNESE) and is an improved version of a recent method called Nonnegative Embedding and Spectral Embedding (NESE) [11]. It inherits the advantages of the NESE method and integrates two different constraints (i.e., an orthogonality constraint and a smoothing constraint) over the nonnegative embedding matrix obtained by NESE, making it more precise.

In Chapter 8, a new method called Multi-view Spectral Clustering with a Self-Taught Robust Graph

Learning (MCSRGL) was introduced. The main innovation of this method is the integration of an additional view formed by the label space. By using the cluster-label correlation, an additional graph is created in addition to the graphs associated with the data space. Moreover, this method imposes a smoothing constraint on the nonnegative embedding matrix \mathbf{H} in its criterion.

Finally, two new methods are proposed in Chapter 9. The first method is called One Step Multi-view Clustering via Consensus Graph Learning and Soft Clustering Assignments (OSMGSCA). It computes the consensus similarity matrix, the similarity matrices of all views, the corresponding spectral projection matrices, and a nonnegative embedding matrix that is used for the final clustering result. By using this paradigm, the effects of noisy affinity matrices can be reduced. In addition, an improvement in the objective function of the OSMGSCA method leads to a new version based on unified matrices that achieves higher performance. This new version is called Unified Multi-view Clustering via Joint Graph Learning and Nonnegative Matrix Assignments (U-MCJGLNMA). Unlike most previous approaches, our methods are based on the idea of multi-view consensus clustering and integrate the representation of all views with cluster assignment into a single framework. To improve the overall performance, these two approaches also apply a number of constraints on the cluster index matrix, such as the cluster label smoothing constraint. Extensive experiments on real and synthetic datasets of different sizes and types show that all of our proposed methods outperform many competing state-of-the-art multi-view clustering methods.

10.2 Perspectives

As an outlook, we envision the development of a scalable variant of the proposed approaches capable of handling large datasets with reasonable computational cost. Moreover, following the CSNE method in [66], our methods can be extended to estimate two nonnegative embedding matrices: the joint and the specific nonnegative embedding matrices, instead of estimating only a single nonnegative matrix. Moreover, following the CNESE method in [97], our approaches can be tested by adding multiple constraints to the nonnegative embedding matrix to make it more precise. By applying multiple constraints to the nonnegative embedding matrix generated by our methods, we can improve the results. We also propose to adapt the proposed methods to other areas of machine learning, including semi-supervised learning and classification problems. Moreover, all our approaches are classical machine learning methods. We can adapt some of these methods to deep variants.

Finally, our methods can be extended to cases where some views have missing data, i.e., the corresponding value in some similarity matrices is missing.

Publications

During this thesis, we were able to publish or submit several papers to international journals and conferences. This chapter contains a list of the accepted papers completed during this thesis.

11.1 International Journals:

1. El Hajjar, S., Dornaika, F., Abdallah, F., and Barrena, N., "Consensus graph and spectral representation for one-step multi-view kernel based clustering." *Knowledge-Based Systems*, 241:108250, 2022.
2. El Hajjar, S., Dornaika, F., and Abdallah, F., "Multi-view Spectral Clustering via Constrained Nonnegative Embedding." *Information Fusion*, 78:209–217, 2022.
3. El Hajjar, S., Dornaika, F., and Abdallah, F., "One-step Multi-view Spectral Clustering with Cluster Label Correlation Graph." *Information Sciences*, 592:97–111, 2022.
4. Dornaika, F., El Hajjar, S., and Abdallah, F., "Direct Multi-view Spectral Clustering with Consistent Kernelized Graph and Convolved Nonnegative Representation." Currently submitted to *Artificial Intelligence Review*.

11.2 International Conferences:

1. El Hajjar, S., Dornaika, F., and Abdallah, F., "Multi-view Spectral Clustering via Integrating Label and Data Graph Learning". In *International Conference on Image Analysis and Processing*, pages 109–120. Springer, 2022.

2. El Hajjar, S., Dornaika, F., and Abdallah, F., "Detection of COVID-19 in X-Ray Images Using Constrained Multi-view Spectral Clustering". SADASC2022 conference.

Bibliography

- [1] Surya Priy. *Clustering in Machine Learning*. GeeksforGeeks, Noida, Uttar Pradesh - 201305, 23 Feb, 2020. See pages [xi](#) and [2](#).
- [2] Pietro Parodi. Computational intelligence with applications to general insurance: a review: I—the role of statistical learning. *Annals of Actuarial Science*, 6(2):307–343, 2012. See pages [xi](#) and [16](#).
- [3] Keshav Dhandhanian and Savan Visalpara. An intuitive introduction to gradient descent. See pages [xi](#) and [29](#).
- [4] Aminah Mardiyah Rufai. Linear regression from scratch pt2: The gradient descent algorithm, May 2021. See pages [xi](#) and [30](#).
- [5] Artificial intelligence (ai). See pages [xi](#) and [35](#).
- [6] MK Gurucharan. Basic cnn architecture: Explaining 5 layers of convolutional neural network, December 2020. See pages [xi](#) and [39](#).
- [7] Matti Pietikäinen. Local binary patterns. See pages [xi](#) and [40](#).
- [8] Feature descriptor: Hog descriptor tutorial, May 2020. See pages [xi](#) and [41](#).
- [9] Vihar Kurama. A guide to densenet, resnext, and shufflenet v2, Apr 2021. See pages [xi](#), [42](#), and [43](#).
- [10] Michael C Thrun and Alfred Ultsch. Clustering benchmark datasets exploiting the fundamental clustering problems. *Data in Brief*, 30:105501, 2020. See pages [xii](#), [45](#), and [64](#).
- [11] Zhanxuan Hu, Feiping Nie, Rong Wang, and Xuelong Li. Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding. *Information Fusion*, 55:251–259, 2020. See pages [6](#), [45](#), [48](#), [54](#), [57](#), [65](#), [67](#), [74](#), [76](#), [78](#), [88](#), [92](#), [94](#), [95](#), [96](#), [101](#), [105](#), [112](#), [113](#), [114](#), [121](#), [126](#), [132](#), [140](#), [142](#), and [151](#).
- [12] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. See pages [18](#), [45](#), [78](#), and [140](#).

- [13] 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. See pages [30](#), [45](#), [76](#), [78](#), [92](#), [94](#), [103](#), [105](#), [115](#), [121](#), [140](#), and [142](#).
- [14] Chao Yang, Zhenwen Ren, Quansen Sun, Mingna Wu, Maowei Yin, and Yuan Sun. Joint correntropy metric weighting and block diagonal regularizer for robust multiple kernel subspace clustering. *Information Sciences*, 500:48–66, 2019. See page [31](#).
- [15] Kun Zhan, Changqing Zhang, Junpeng Guan, and Junsheng Wang. Graph learning for multiview clustering. *IEEE Transactions on Cybernetics*, 48(10):2887–2895, 2017. See pages [45](#), [92](#), [94](#), [105](#), [121](#), [140](#), and [142](#).
- [16] Yu-Meng Xu, Chang-Dong Wang, and Jian-Huang Lai. Weighted multi-view clustering with feature selection. *Pattern Recognition*, 53:25–35, 2016. See pages [45](#), [46](#), and [47](#).
- [17] Linlin Zong, Xianchao Zhang, Xinyue Liu, and Hong Yu. Weighted multi-view spectral clustering based on spectral perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. See page [45](#).
- [18] Tian Xia, Dacheng Tao, Tao Mei, and Yongdong Zhang. Multiview spectral embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(6):1438–1446, 2010. See pages [45](#) and [46](#).
- [19] Chang Xu, Dacheng Tao, and Chao Xu. Multi-view self-paced learning for clustering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. See page [45](#).
- [20] Feiping Nie, Lai Tian, and Xuelong Li. Multiview clustering via adaptively weighted procrustes. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2022–2030, 2018. See pages [45](#), [47](#), [54](#), [76](#), [78](#), [92](#), [94](#), [105](#), [121](#), [140](#), and [142](#).
- [21] Mouxing Yang, Yunfan Li, Peng Hu, Jinfeng Bai, Jian Cheng Lv, and Xi Peng. Robust multi-view clustering with incomplete information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. See page [45](#).
- [22] Xiaochuang Shu, Xiangdong Zhang, and Qianqian Wang. Self-weighted graph learning for multi-view clustering. *Neurocomputing*, 2022. See page [45](#).
- [23] Shudong Huang, Zhao Kang, and Zenglin Xu. Auto-weighted multi-view clustering via deep matrix decomposition. *Pattern Recognition*, 97:107015, 2020. See pages [45](#), [46](#), and [115](#).
- [24] Shudong Huang, Zhao Kang, Ivor W Tsang, and Zenglin Xu. Auto-weighted multi-view clustering via kernelized graph learning. *Pattern Recognition*, 88:174–184, 2019. See pages [45](#), [48](#), [54](#), [57](#), [67](#), [68](#), [69](#), [70](#), [74](#), [76](#), [78](#), [88](#), [92](#), [94](#), [95](#), [96](#), [103](#), [105](#), [115](#), [121](#), [140](#), and [142](#).
- [25] Pengzhen Ren, Yun Xiao, Pengfei Xu, Jun Guo, Xiaojiang Chen, Xin Wang, and Dingyi Fang. Robust auto-weighted multi-view clustering. In *IJCAI*, pages 2644–2650, 2018. See page [45](#).

- [26] Zhenni Jiang and Xiyu Liu. Adaptive knn and graph-based auto-weighted multi-view consensus spectral learning. *Information Sciences*, 609:1132–1146, 2022. See page 45.
- [27] Zongze Wu, Sihui Liu, Chris Ding, Zhigang Ren, and Shengli Xie. Learning graph similarity with large spectral gap. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019. See pages 45 and 47.
- [28] Martha White, Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Convex multi-view subspace learning. In *Nips*, pages 1682–1690. Lake Tahoe, Nevada, 2012. See page 45.
- [29] Guang-Yu Zhang, Xiao-Wei Chen, Yu-Ren Zhou, Chang-Dong Wang, Dong Huang, and Xiao-Yu He. Kernelized multi-view subspace clustering via auto-weighted graph learning. *Applied Intelligence*, 52(1):716–731, 2022. See page 45.
- [30] Derek Greene and Pádraig Cunningham. A matrix factorization approach for integrating multiple data views. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 423–438. Springer, 2009. See pages 45 and 48.
- [31] 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. See pages 45 and 48.
- [32] Z. Yang, Y. Zhang, Y. Xiang, W. Yan, and S. Xie. Non-negative matrix factorization with dual constraints for image clustering. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(7):2524–2533, 2020. See pages 45 and 49.
- [33] Wei Zhu, Feiping Nie, and Xuelong Li. Fast spectral clustering with efficient large graph construction. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2492–2496, 2017. See pages 45, 92, 94, and 142.
- [34] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML’11*, page 393–400, Madison, WI, USA, 2011. See pages 45, 46, 53, 76, 78, 92, 94, 105, 121, 139, and 142.
- [35] Abhishek Kumar, Piyush Rai, and Hal Daumé. Co-regularized multi-view spectral clustering. In *Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS’11*, page 1413–1421, Red Hook, NY, USA, 2011. See pages 45, 76, 78, 92, 94, 105, 139, and 142.
- [36] Yujiao Zhao, Yu Yun, Xiangdong Zhang, Qin Li, and Quanxue Gao. Multi-view spectral clustering with adaptive graph learning and tensor Schatten p-norm. *Neurocomputing*, 468:257–264, 2022. See page 45.
- [37] Haizhou Yang, Quanxue Gao, Wei Xia, Ming Yang, and Xinbo Gao. Multi-view spectral clustering with bipartite graph. *IEEE Transactions on Image Processing*, 2022. See page 45.

- [38] Abhishek Kumar, Piyush Rai, and Hal Daumé. Co-regularized multi-view spectral clustering. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. See pages [46](#), [53](#), and [121](#).
- [39] Krishna Kumar Sharma and Ayan Seal. Multi-view spectral clustering for uncertain objects. *Information Sciences*, 547:723–745, 2021. See page [46](#).
- [40] Zongmo Huang, Yazhou Ren, Xiaorong Pu, Lili Pan, Dezhong Yao, and Guoxian Yu. Dual self-paced multi-view clustering. *Neural Networks*, 140:184–192, 2021. See page [46](#).
- [41] Chang Tang, Xinzhong Zhu, Xinwang Liu, Miaomiao Li, Pichao Wang, Changqing Zhang, and Lizhe Wang. Learning a joint affinity graph for multiview subspace clustering. *IEEE Transactions on Multimedia*, 21(7):1724–1736, 2018. See page [46](#).
- [42] Xiaofeng Zhu, Shichao Zhang, Yonghua Zhu, Wei Zheng, and Yang Yang. Self-weighted multi-view fuzzy clustering. *ACM transactions on knowledge discovery from data (TKDD)*, 14(4):1–17, 2020. See page [46](#).
- [43] Yazhou Ren, Shudong Huang, Peng Zhao, Minghao Han, and Zenglin Xu. Self-paced and auto-weighted multi-view clustering. *Neurocomputing*, 383:248–256, 2020. See pages [46](#) and [115](#).
- [44] Feiping Nie, Guohao Cai, Jing Li, and Xuelong Li. Auto-weighted multi-view learning for image clustering and semi-supervised classification. *IEEE Transactions on Image Processing*, 27(3):1501–1511, 2017. See pages [46](#), [54](#), [76](#), [78](#), [92](#), [94](#), [105](#), [115](#), [121](#), [139](#), and [142](#).
- [45] Juncheng Lv, Zhao Kang, Boyu Wang, Luping Ji, and Zenglin Xu. Multi-view subspace clustering via partition fusion. *Information Sciences*, 560:410–423, 2021. See page [47](#).
- [46] Guang-Yu Zhang, Yu-Ren Zhou, Chang-Dong Wang, Dong Huang, and Xiao-Yu He. Joint representation learning for multi-view subspace clustering. *Expert Systems with Applications*, 166:113913, 2021. See page [47](#).
- [47] Yongyong Chen, Xiaolin Xiao, Chong Peng, Guangming Lu, and Yicong Zhou. Low-rank tensor graph learning for multi-view subspace clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021. See page [47](#).
- [48] Aparajita Khan and Pradipta Maji. Multi-manifold optimization for multi-view subspace clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. See page [47](#).
- [49] Mengjing Sun, Siwei Wang, Pei Zhang, Xinwang Liu, Xifeng Guo, Sihang Zhou, and En Zhu. Projective multiple kernel subspace clustering. *IEEE Transactions on Multimedia*, 24:2567–2579, 2021. See page [47](#).
- [50] Xiaoqian Zhang, Zhenwen Ren, Huaijiang Sun, Keqiang Bai, Xinghua Feng, and Zhigui Liu. Multiple kernel low-rank representation-based robust multi-view subspace clustering. *Information Sciences*, 551:324–340, 2021. See page [47](#).

- [51] Peng Chen, Liang Liu, Zhengrui Ma, and Zhao Kang. Smoothed multi-view subspace clustering. In *International Conference on Neural Computing for Advanced Applications*, pages 128–140. Springer, 2021. See page 47.
- [52] Changqing Zhang, Qinghua Hu, Huazhu Fu, Pengfei Zhu, and Xiaochun Cao. Latent multi-view subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4279–4287, 2017. See page 47.
- [53] Guoli Niu, Youlong Yang, and Liqin Sun. One-step multi-view subspace clustering with incomplete views. *Neurocomputing*, 438:290–301, 2021. See page 47.
- [54] 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. See page 48.
- [55] Zhenwen Ren and Quansen Sun. Simultaneous global and local graph structure preserving for multiple kernel clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1839–1851, 2021. See pages 48 and 136.
- [56] Yilu Zheng, X. Zhang, Yinlong Xu, Mingwei Qin, Zhenwen Ren, and Xuqian Xue. Robust multi-view subspace clustering via weighted multi-kernel learning and co-regularization. *IEEE Access*, 8:113030–113041, 2020. See page 48.
- [57] W. Guo, Y. Shi, and S. Wang. A unified scheme for distance metric learning and clustering via rank-reduced regression. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12, 2019. See page 49.
- [58] X. Wang, T. Zhang, and X. Gao. Multiview clustering based on non-negative matrix factorization and pairwise measurements. *IEEE Transactions on Cybernetics*, 49(9):3333–3346, 2019. See page 49.
- [59] Hao Wang, Yan Yang, and Bing Liu. Gmc: Graph-based multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1116–1129, 2020. See page 49.
- [60] D. Huang, C. D. Wang, H. Peng, J. Lai, and C. K. Kwoh. Enhanced ensemble clustering via fast propagation of cluster-wise similarities. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1):508–520, 2021. See page 49.
- [61] Deyan Xie, Xiangdong Zhang, Quanxue Gao, Jiale Han, Song Xiao, and Xinbo Gao. Multiview clustering by joint latent representation and similarity learning. *IEEE Transactions on Cybernetics*, 50(11):4848–4854, 2020. See page 49.
- [62] 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. See pages 49 and 56.
- [63] K. Zhan, C. Zhang, J. Guan, and J. Wang. Graph learning for multiview clustering. *IEEE Transactions on Cybernetics*, 48(10):2887–2895, 2018. See pages 49, 56, 76, and 78.

- [64] Xiaofeng Zhu, Shichao Zhang, Wei He, Rongyao Hu, Cong Lei, and Pengfei Zhu. One-step multi-view spectral clustering. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):2022–2034, 2018. See pages [50](#) and [54](#).
- [65] Zhenwen Ren, Haoran Li, Chao Yang, and Quansen Sun. Multiple kernel subspace clustering with local structural graph and low-rank consensus kernel learning. *Knowledge-Based Systems*, 188:105040, 2020. See page [50](#).
- [66] Hongwei Yin, Wenjun Hu, Fanzhang Li, and Jungang Lou. One-step multi-view spectral clustering by learning common and specific nonnegative embeddings. *International Journal of Machine Learning and Cybernetics*, 12(7):2121–2134, 2021. See pages [50](#), [55](#), and [152](#).
- [67] Man-Sheng Chen, Ling Huang, Chang-Dong Wang, and Dong Huang. Multi-view clustering in latent embedding space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3513–3520, 2020. See pages [50](#), [55](#), [92](#), [94](#), [95](#), [96](#), [121](#), [140](#), and [142](#).
- [68] L. Xing, B. Chen, S. Du, Y. Gu, and N. Zheng. Correntropy-based multiview subspace clustering. *IEEE Transactions on Cybernetics*, pages 1–14, 2019. See page [50](#).
- [69] Xi Peng, Zhenyu Huang, Jiancheng Lv, Hongyuan Zhu, and Tianyi Joey Zhou. Comic: Multi-view clustering without parameter selection. *international conference on machine learning*, 2019. See page [50](#).
- [70] Hao Wang, Linlin Zong, Bing Liu, Yan Yang, and Wei Zhou. Spectral perturbation meets incomplete multi-view data. In *International Joint Conference on Artificial Intelligence*, pages 3677–3683, 2019. See page [50](#).
- [71] Guoqing Chao, Jiangwen Sun, Jin Lu, An-Li Wang, Daniel D Langleben, Chiang-Shan Li, and Jinbo Bi. Multi-view cluster analysis with incomplete data to understand treatment effects. *Information Sciences*, 494:278–293, 2019. See page [51](#).
- [72] S. Jing-Tao and Z. Qiu-Yu. Completion of multiview missing data based on multi-manifold regularised non-negative matrix factorisation. *Artificial Intelligence Review*, 53:5411–5428, 2020. See page [51](#).
- [73] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. See page [51](#).
- [74] Mingjing Du, Shifei Ding, and Hongjie Jia. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Systems*, 99:135–145, 2016. See page [51](#).
- [75] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014. See page [51](#).
- [76] T. Zhou, Changqing Zhang, Xi Peng, H. Bhaskar, and J. Yang. Dual shared-specific multiview subspace clustering. *IEEE Transactions on Cybernetics*, 50:3517–3530, 2020. See page [52](#).

- [77] Zhanxuan Hu, Feiping Nie, Wei Chang, Shuzheng Hao, Rong Wang, and Xuelong Li. Multi-view spectral clustering via sparse graph learning. *Neurocomputing*, 384:1–10, 2020. See pages [52](#), [55](#), [76](#), [78](#), [92](#), [94](#), [95](#), [96](#), [140](#), and [142](#).
- [78] Mitsuhiro Horie and Hiroyuki Kasai. Consistency-aware and inconsistency-aware graph-based multi-view clustering. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1472–1476. IEEE, 2021. See pages [52](#), [56](#), [76](#), [78](#), [92](#), [94](#), [95](#), [96](#), [140](#), and [142](#).
- [79] Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 129–136, 2009. See page [52](#).
- [80] Xiao Yu, Hui Liu, Yan Wu, and Caiming Zhang. Fine-grained similarity fusion for multi-view spectral clustering. *Information Sciences*, 568:350–368, 2021. See pages [52](#) and [56](#).
- [81] Hao Cai, Bo Liu, Yanshan Xiao, and LuYue Lin. Semi-supervised multi-view clustering based on orthonormality-constrained nonnegative matrix factorization. *Information Sciences*, 536:171–184, 2020. See pages [52](#) and [55](#).
- [82] Changan Yuan, Yonghua Zhu, Zhi Zhong, Wei Zheng, and Xiaofeng Zhu. Robust self-tuning multi-view clustering. *World Wide Web*, 25(2):489–512, 2022. See page [53](#).
- [83] Guoqing Chao, Shiliang Sun, and Jinbo Bi. A survey on multi-view clustering. *IEEE Transactions on Artificial Intelligence*, 2021. See page [53](#).
- [84] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Mining and Analytics*, 1(2):83–107, 2018. See page [53](#).
- [85] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-100). 1996. See page [59](#).
- [86] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994. See page [59](#).
- [87] Amir Monadjemi, BT Thomas, and Majid Mirmehdi. Experiments on high resolution images towards outdoor scene classification. Technical report. See page [60](#).
- [88] John Winn and Nebojsa Jojic. Locus: Learning object classes with unsupervised segmentation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 756–763. IEEE, 2005. See page [60](#).
- [89] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009. See page [60](#).

- [90] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 377–384, 2006. See page [60](#).
- [91] 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. See page [61](#).
- [92] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. See page [61](#).
- [93] F. Nie, X. Wang, M. Jordan, and H. Huang. The constrained laplacian rank algorithm for graph-based clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016. See pages [65](#), [73](#), [92](#), [104](#), [118](#), [134](#), and [137](#).
- [94] M. T. Kejani, F. Dornaika, and H. Talebi. Graph convolution networks with manifold regularization for semi-supervised learning. *Neural Networks*, 2020. See page [70](#).
- [95] Feiping Nie, Jing Li, Xuelong Li, et al. Self-weighted multiview clustering with multiple graphs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2564–2570, 2017. See pages [76](#), [78](#), [92](#), [94](#), [103](#), [105](#), [121](#), [139](#), and [142](#).
- [96] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Affinity aggregation for spectral clustering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 773–780, 2012. See pages [76](#), [78](#), [92](#), [94](#), [105](#), [121](#), [140](#), and [142](#).
- [97] Sally El Hajjar, Fadi Dornaika, and Fahed Abdallah. Multi-view spectral clustering via constrained nonnegative embedding. *Information Fusion*, 2021. See pages [76](#), [78](#), [92](#), [94](#), [95](#), [96](#), [140](#), [142](#), and [152](#).
- [98] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. See pages [84](#), [97](#), [125](#), and [148](#).
- [99] F. Dornaika, A. Baradaaji, and Y. El Traboulsi. Semi-supervised classification via simultaneous label and discriminant embedding estimation. *Information Sciences*, 546:146 – 165, 2021. See page [103](#).
- [100] Shaojun Shi, Feiping Nie, Rong Wang, and Xuelong Li. Auto-weighted multi-view clustering via spectral embedding. *Neurocomputing*, 2020. See page [103](#).
- [101] Guangfeng Lin, Kaiyang Liao, Bangyong Sun, Yajun Chen, and Fan Zhao. Dynamic graph fusion label propagation for semi-supervised multi-modality classification. *Pattern Recognition*, 68:14–23, 2017. See page [116](#).
- [102] Bo Wang, Zhuowen Tu, and John K Tsotsos. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *Proceedings of the IEEE international conference on computer vision*, pages 425–432, 2013. See page [116](#).
- [103] Saeedeh Bahrami, Alireza Bosaghzadeh, and Fadi Dornaika. Multi similarity metric fusion in graph-based semi-supervised learning. *Computation*, 7(1):15, 2019. See page [116](#).

- [104] Guang-Yu Zhang, Yu-Ren Zhou, Xiao-Yu He, Chang-Dong Wang, and Dong Huang. One-step kernel multi-view subspace clustering. *Knowledge-Based Systems*, 189:105126, 2020. See page [130](#).
- [105] Xiaofeng Zhu, Shichao Zhang, Wei He, Rongyao Hu, Cong Lei, and Pengfei Zhu. One-step multi-view spectral clustering. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):2022–2034, 2019. See page [136](#).
- [106] Kun Zhan, Feiping Nie, Jing Wang, and Yi Yang. Multiview consensus graph clustering. *IEEE Transactions on Image Processing*, 28(3):1261–1270, 2019. See page [140](#).
- [107] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. See page [140](#).