CLUSTERING AND SEMI-SUPERVISED CLASSIFICATION WITH APPLICATION TO DRIVER DISTRACTION DETECTION

LIU TIANCHI

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

2018

Acknowledgements

I am very grateful to my supervisor Professor Huang Guang-Bin for his support and excellent supervision. He sees the potential in me and has always been the source of inspiration. I would like to thank my co-supervisor Associate Professor Lin Zhiping for his patient guidance and kind encouragement. He spent a lot of time on helping me find my way to become a researcher.

I am thankful to Dr. Huang Gao, Dr. Yang Yan, and Dr. Liyanaarachchi Lekamalage Chamara Kasun, with whom I have worked closely, for their expertise and thoughtful discussions. Exchanging research ideas with them has been one of the most enjoyable parts of the research study. Special thanks go to Dr. Yang Yan for letting me work on the valuable data set she collected during her Ph.D. study.

I would like to thank the technical support from Research Staff Office, BMW-NTU Future Mobility Research Lab, and Delta-EEE Joint Research Lab in School of Electrical and Electronic Engineering, Nanyang Technological University. It is impossible to list all my colleagues and lab mates; Here let me thank some of them: Dr. Sun Haoqi, Dr. Guo Hongliang, Dr. Cao Zhiguang, Mr. Han Wei, Mr. Cui Dongshun, Mr. Li Yue, and Mr. Song Kang. The interesting conversations, celebration gatherings, and encouraging chats have made my Ph.D. study very memorable.

Last but not the least, I would like to thank my parents for their love, understanding and support throughout these years.

Table of Contents

Summary

Clustering and Semi-Supervised Classification (SSC) algorithms can make use of unlabeled training data and thus have the potential to alleviate labeling costs. For example, Extreme Learning Machine (ELM) was recently extended to semi-supervised learning and clustering with promising performance. Meanwhile, it is either costly or infeasible to obtain labeled training samples in some real-world applications. The thesis investigates clustering and SSC algorithms with application to driver distraction detection.

Firstly, the thesis investigates embedding-based clustering. The desirable properties of embedding are reviewed in the literature, e.g., preserving the intrinsic data structure and maximizing the class separability. To obtain better embedding for clustering, the thesis considers both properties together and develops a novel clustering algorithm referred to as ELM for Joint Embedding and Clustering (ELM-JEC). Experimental studies on a wide range of benchmark datasets have show that ELM-JEC is competitive with the related methods.

Secondly, the thesis investigates graph-based clustering. One limitation of existing graph learning methods is that they adjust the graph based on either the original data or the linearly projected data, which may not effectively reveal the underlying low-dimensional structures. To address this limitation, this thesis develops dual data representations, i.e., the original data and their nonlinear embedding obtained via an ELM-based neural network, and uses them as the basis for graph learning. The resulting algorithm is named as clustering based on ELM and Constrained Laplacian Rank (ELM-CLR). The experimental results show that ELM-CLR outperforms other adaptive graph learning methods on most benchmark datasets.

Finally, the thesis applies the proposed clustering algorithms, i.e., ELM-JEC and ELM-CLR, and several SSC algorithms to driver distraction detection. The clustering algorithms are used on unlabeled data to generate preliminary labels as reference to assist human experts in the labeling process. In terms of the clustering accuracy, both proposed clustering algorithms perform better or on par with the related algorithms. The best clustering accuracy is achieved by ELM-JEC. Moreover, the research question of "which type of SSC method is more suitable for driver distraction detection?" is answered by evaluating two popular types of semi-supervised methods on a real-world dataset of drivers' eye and head movements. The experimental results show that the graph-based methods achieve twice the improvement by the low-density-separation based method. It has also been shown that 1) the graph-based methods reduce the required amount of labeled training data, and 2) the benefits in detection accuracy increase with the size of unlabeled datasets.

Overall, the thesis contributes two novel clustering algorithms by making use of ELM-based embedding and discovers that 1) better clustering performance on some datasets is expected, if the embedding preserves the intrinsic local structure and maximizes the class separability simultaneously, and 2) Both original and nonlinear embedded spaces are crucial to learning graphs with clear clusters. Moreover, the thesis contributes to the research on driver distraction detection by putting forward a semi-supervised driver distraction detection system with efficient labeling assistance and verifies it on an on-road driver distraction dataset.

List of Figures

List of Tables

List of Acronyms and Symbols

Acronyms

C-ELM-AE	Clustering with ELM-AE
C-ELM-FS	Clustering in Extreme Learning Machine Feature Space
CAN	Clustering with Adaptive Neighbors
CLR	Constrained Laplacian Rank
DEC	Discriminative Embedded Clustering
ELM	Extreme Learning Machine
ELM-CLR	clustering based on Extreme Learning Machine and Constrained Laplacian Rank
ELM-DC	ELM-based Discriminative Clustering
ELM-JEC	Extreme Learning Machine for Joint Embedding and Clustering
ELMC^{LDA}	Extreme Learning Machine Clustering based on Linear Discriminant Analysis
FN	False Negative
FP	False Positive
LapSVM	Laplacian Support Vector Machine

LDA	Linear Discriminant Analysis
LDA-KM	LDA-guided k-Means clustering
NCut	Normalized Cut
PCA	Principal Component Analysis
PCAN	Projected Clustering with Adaptive Neighbors
S^3VM	Semi-Supervised Support Vector Machine
SBN-SC	Static Bayesian Network with Supervised Clustering
SC	Spectral Clustering
SLFNs	Single hidden Layer Feedforward neural Networks
SS-ELM	Semi-Supervised Extreme Learning Machine
SSC	Semi-Supervised Classification
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TSVM	Transductive Support Vector Machine
UD-ELM	Unsupervised Discriminative Extreme Learning Machine
US-ELM	UnSupervised Extreme Learning Machine

Symbols

$c \in \mathbb{Z}$	Number of clusters
$n \in \mathbb{Z}$	Number of data samples
$n_I \in \mathbb{Z}$	Dimensionality of the input space of ELM

$n_H \in \mathbb{Z}$	Dimensionality of the hidden neuron space of ELM
$n_O \in \mathbb{Z}$	Dimensionality of the output space of ELM
$\boldsymbol{x} \in \mathbb{R}^{n_I}$	Data sample in the original feature space
$\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^{1\times n_H}$	Output of ELM hidden layer with respect to a data sample $\boldsymbol{x}$
$\boldsymbol{y} \in \mathbb{R}^{1\times n_O}$	Data sample in the embedded space
$\boldsymbol{f} \in \{0,1\}^c$	Cluster indicator
$\boldsymbol{g} \in \mathbb{R}^d$	Cluster centroid in d-dimensional space
$\boldsymbol{\beta} \in \mathbb{R}^{n_H\times n_O}$	Output weights of ELM
$\boldsymbol{X} \in \mathbb{R}^{n\times n_I}$	Data matrix in the original feature space
$\bar{\boldsymbol{X}} \in \mathbb{R}^{n\times n_I}$	Mean-centred data matrix in the original feature space
$\boldsymbol{H} \in \mathbb{R}^{n\times n_H}$	Output matrix of ELM hidden layer
$\bar{\boldsymbol{H}} \in \mathbb{R}^{n\times n_H}$	Mean-centred output matrix of ELM hidden layer
$\boldsymbol{F} \in \{0,1\}^{n\times c}$	Cluster indicator matrix
$\boldsymbol{G} \in \mathbb{R}^{d\times c}$	Cluster centroid matrix in d-dimensional space with c clusters
$\boldsymbol{W} \in \mathbb{R}^{n\times n}$	Similarity matrix of a graph
$\boldsymbol{L} \in \mathbb{R}^{n\times n}$	Graph Laplacian matrix
$\boldsymbol{S}_w \in \mathbb{R}^{d\times d}$	Within-cluster scatter matrix of d-dimensional data
$\boldsymbol{S}_b \in \mathbb{R}^{d\times d}$	Between-cluster scatter matrix of d-dimensional data
$\boldsymbol{S}_t \in \mathbb{R}^{d\times d}$	Total scatter matrix of d-dimensional data
$\boldsymbol{S}_t^H \in \mathbb{R}^{n_H\times n_H}$	Total scatter matrix of data in ELM hidden neuron space

$\boldsymbol{P} \in \mathbb{R}^{n_I \times n_O}$	Linear transformation matrix from n_I-dimensional space to n_O-dimensional space
$\boldsymbol{I}_d \in \mathbb{R}^{d \times d}$	The identity matrix of size d
$\mathbf{1}_d \in \mathbb{R}^d$	A d-dimensional vector with all elements equal to 1

Chapter 1

Introduction

Chapter 1 begins with the background and motivation of studying clustering and semi-supervised classification as well as the application to driver distraction detection. Then this chapter identifies the objectives of the thesis and highlights the contributions. At last, the structure of the thesis is presented.

1.1 Background and Motivation

The history of humans' attempt to classification dates back to the origin of human language, when people distinguished edible and poisonous plants in order to communicate past experience to the family [1]. It has always been essential to categorize objects into groups based on the relevant information, instead of regarding every object as a distinct, unrelated entity or treating them all as the same. In machine learning, *clustering* is the unsupervised classification of patterns (observations or data points) into groups (clusters or classes) so that the data points in the same clusters are more similar to each other than those in different clusters [2]. Clustering has a wide variety of applications in engineering, computer science, biology, social science, geology, and marketing [2, 3, 4].

Supervised classification is the task of identifying to which of a set of classes a new data point belongs on the basis of a training set containing observations and their category (label) information. In some cases, not all observations come with label information. The observations that come with label information are known as *labeled training data*, while the ones with no label information are *unlabeled training data*. *Semi-supervised classification* algorithms are able to learn based on a combination of both labeled and unlabeled training data. Semi-supervised classification draws increasing interests from machine learning research community [5].

Many real-world applications face the problem of scarce label information and need clustering and semi-supervised classification. In the fields of text classification [6] and image classification [7], semi-supervised methods have shown promising performance. Taking the driver distraction detection [8, 9, 10] as an example: the collection of positively labeled training data can be unsafe and not user friendly, because it requires the driver to drive under distraction; even though there are abundant training data available in naturalistic driving record, labeling is very costly, because it requires additional data sources (such as driver face video) and human experts. Therefore, for applications like driver distraction detection, it is quite necessary to reduce the labeling cost and make better use of the unlabeled training data.

Learning data representation, often referred to as the *embedding*, has shown to improve clustering performance, as data in the original space contain noise and irrelevant information [11, 12]. As a supervised training algorithm for single hidden layer neural networks with the hidden neurons randomly generated and fixed, Extreme Learning Machine (ELM) has the advantage of nonlinear modeling capability, universal approximation capability, and little parameter tuning [13, 14]. Recently, Huang et al. used ELM to generate the embedding of the data and reported improved clustering performance on the embedding [15]. Therefore, it is very promising to further investigate the capability ELM-based embedding so as to achieve better clustering performance.

1.2 Objectives and Contributions

In view of the above-mentioned motivation, the objectives of the thesis are defined as follows:

(1) to improve the performance of clustering algorithms by taking advantage of ELM.

(2) to propose a semi-supervised driver distraction detection system with efficient labeling assistance.

To achieve the first objective, the first study proposes to use the output of ELM as an embedding that satisfies the need of clustering. The study contributes by presenting a comprehensive review of the state-of-the-art nonlinear embedding-based clustering methods. The review summarizes the merits as follows: 1) nonlinearity, 2) maximizing the class separability, 3) preserving intrinsic local structures, 4) simultaneous embedding and clustering. In view of the summarized properties, the study proposes to learn the output weights of ELM and the cluster indicators jointly so as to incorporate all the merits. This method is named as ELM for Joint Embedding and Clustering (ELM-JEC). On a wide range of real-world benchmark dataset, ELM-JEC outperforms the other state-of-the-art clustering methods. The work has been published in a journal with the

following details:

– Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. "Extreme Learning Machine for Joint Embedding and Clustering," *Neurocomputing*, vol. 277, pp. 78-88, 2018.

Towards the first objective, another study is conducted. A review on the state-of-the-art adaptive graph learning methods reveals their limitation: only the original data and the linearly projected data are used to learn the similarity graph, which may not effectively reveal the underlying low-dimensional structures [16, 17]. The thesis addresses the limitation by 1) proposing dual representations, i.e., the original data and the non-linear embedding of the data obtained via an ELM-based neural network, and 2) developing an adaptive graph learning method based on the dual representations. The experimental results demonstrate that the proposed clustering algorithm based on ELM and Constrained Laplacian Rank (ELM-CLR) achieves superior performance compared with existing adaptive graph learning algorithms on a board range of real-world benchmark dataset. This work has been published in a journal with the following details:

– Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. "An Adaptive Graph Learning Method based on Dual Data Representations for Clustering," *Pattern Recognition*, vol. 77, pp. 126-139, 2018.

To achieve the second objective, the thesis answers the research question of "which type of semi-supervised method is more suitable for driver distraction detection?" A semi-supervised driver distraction detection framework is first proposed. Then two popular types of semi-supervised methods are investigated using a real-world dataset. The empirical studies demonstrate that both the graph-based semi-supervised methods and the low-density-separation-based ones can improve the detection performance with the help of additional unlabeled training data, compared with the supervised methods. Moreover, the graph-based semi-supervised methods have shown more promising improvement in

terms of detection accuracy. Based on these observations, the study further investigates several important properties of the proposed system: influence by the amount of unlabeled data, influence by the amount of labeled data, and sensitivity to hyper-parameters. This work has been published in a journal with the following details [18]:

– Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. "Driver Distraction Detection Using Semi-Supervised Machine Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1108-1120, 2016.

Towards the second objective, the two clustering algorithms proposed in the first and second studies were tested on the real-word driver distraction classification dataset to create preliminary labels for experts. The empirical results demonstrate that ELM-JEC and ELM-CLR perform on a par with or better than their competitive methods. The clustering results can be used as a reference for human experts while they are labeling a large amount of data. Since the clustering performance is satisfactory, efforts can be saved because the human experts only need to check the correctness of the labels instead of creating them from scratch.

In the study of driver distraction detection, the desired number of clusters is known to be two, because clustering serves the purpose of labeling assistance and the downstream task requires two classes, i.e., distracted driving state and normal driving state. This PhD work will benefit this type of applications and assumes the number of clusters is known, where ELM-JEC and ELM-CLR can be readily applied.

1.3 Structure of Thesis

The remaining parts of the thesis are organized as follow.

Chapter 2 reviews data, embedding, graph, ELM, and the state-of-the-art clustering and

semi-supervised classification methods, as well as the related work on driver distraction detection.

Chapter 3 investigates the embedding-based clustering algorithms. The merits of embedding-based clustering methods are first analyzed, and then the ELM-JEC algorithm is proposed by combining all the merits and evaluated.

Chapter 4 investigates the graph-based clustering algorithms. The limitation of the state-of-the-art adaptive graph learning algorithms is first analyzed, and then an improved method ELM-CLR is developed based on novel dual representations.

Chapter 5 presents and analyzes the eye and head movement dataset collected from an on-road experiment. Furthermore, it proposes and evaluates a novel semi-supervised driver distraction detection system. Lastly, it evaluates a large range of clustering methods including ELM-JEC and ELM-CLR for efficient labeling assistance.

Chapter 6 concludes the thesis and points out several future research directions.

Chapter 2

Literature Review

Chapter 2 first reviews some background on data, embedding and graph in Section 2.1 and Extreme Learning Machine in Section 2.2. Then an in-depth review of the state-of-the-art clustering and SSC methods are reviewed in Section 2.3 and Section 2.4 respectively. Lastly, the application of driver distraction detection is introduced in Section 2.5 with a focus on the modeling and labeling process.

2.1 Data, Embedding, and Graph

The following notations are used throughout the thesis unless defined otherwise. Sets are written as calligraphic uppercase $\mathcal{A}, \mathcal{B}, \mathcal{C}$, and matrices are written as bold uppercase $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$. Column vectors are written as bold lowercase $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$. Scalers are written as nonbold α, β, γ. $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ represents a real matrix of dimension $m \times n$. $\boldsymbol{x} \in \mathbb{R}^m$ represents a column vector of length m. The transpose of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^T$. $\mathrm{Tr}(\cdot)$ denotes the trace of a square matrix. $\det(\cdot)$ denotes the determinant of a square matrix.The (i, j)-th entry of a matrix $\boldsymbol{A}$ is denoted as a_{ij}. The i-th entry of a vector $\boldsymbol{a}$ is denoted as a_i. $\mathbf{1}_d$ denotes the d-dimensional vector with all elements equal to 1. $\boldsymbol{I}_d$ denotes the identity matrix of dimensionality d. $\|\cdot\|_2$ denotes the L2-norm of a vector:

$$\|\boldsymbol{a}\|_2 = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}. \tag{2.1}$$

$\|\cdot\|_F$ and $\|\cdot\|_1$ denote the Frobenius and L1-norm (a.k.a. the maximum absolute column sum norm) of a matrix, respectively:

$$\begin{aligned} \|\boldsymbol{A}\|_F &= \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}, \\ \|\boldsymbol{A}\|_1 &= \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|. \end{aligned} \tag{2.2}$$

The training data are usually denoted as $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ represents a sample and each entry of $\boldsymbol{x}_i$ represents a variable/feature. The embedding of the data is usually denoted as $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^n$, where $\boldsymbol{y}_i \in \mathbb{R}^d$ is the new data representation of dimension d corresponding to the i-th training data $\boldsymbol{x}_i$. In this way, each sample is represented independently. The mathematical structures used to capture the pairwise relationship between samples are graphs. Each graph is defined in terms of three components: vertex, edge, and weight. Therefore, a undirected weighted graph can be represented using its vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and weight set $\mathcal{W}$ as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$.

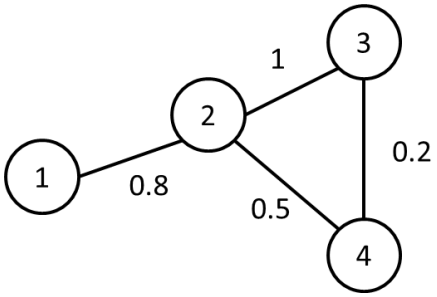


Figure 2.1: Illustration of a undirected weighted graph.

Figure 2.1 shows a simple illustration of a undirected weighted graph, where $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$, $\mathcal{E} = \{e_1(v_1, v_2), e_2(v_2, v_3), e_3(v_2, v_4), e_4(v_3, v_4)\}$, $\mathcal{W} = \{w_1 = 0.8, w_2 = 1, w_3 = 0.5, w_4 = 0.2\}$. In this thesis, the weight information is represented using a weight matrix $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{n \times n}$, where n is the number of vertices. In this example, the weight matrix is

$$\boldsymbol{W} = \begin{bmatrix} 1 & 0.8 & 0 & 0 \\ 0.8 & 1 & 1 & 0.5 \\ 0 & 1 & 1 & 0.2 \\ 0 & 0.5 & 0.2 & 1 \end{bmatrix}. \tag{2.3}$$

Since the edge information can be directly implied from the weight matrix, i.e., there is an edge between two vertices if the weight is not zero, the edge set can be omitted.

Given a training data set $\mathcal{X}$, an important question is how to represent the pair-wise similarity relationship. Let $\mathcal{G} = \{\mathcal{X}, \mathcal{W}\}$ be an undirected weighted graph with vertex set $\mathcal{X}$. The weight set $\mathcal{W}$ can be represented by a weight matrix $\boldsymbol{W} = [w_{ij}]$. Since each entry of the weight matrix represents the similarity between the two vertices, the weight matrix is also called the *similarity matrix* and can be constructed in two steps [19]:

- **Determining the Edge.** An edge should be created if two samples are considered "similar." For example, the most common practise is "k nearest neighbor" method: vertices i and j are connected if i is among k nearest neighbors of j or j is among k nearest neighbors of i (k is a parameter). Another common way is "ϵ

neighborhood" method: vertices i and j are connected if their Euclidean distance is smaller than ϵ (ϵ is a parameter).

- **Choosing the Weight.** The weight should be proportional to the similarity between two vertices. One way of choosing the weight is to let $w_{ij} = 1$ if there is an edge between vertices i and j and $w_{ij} = 0$ otherwise. Another common way is to use the heat kernel. If vertice i and j are connected, choose the weight

$$w_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{t}\right), \tag{2.4}$$

otherwise, $w_{ij} = 0$, where $t \in \mathbb{R}$ is a parameter.

Given the similarity matrix $\boldsymbol{W}$ of a graph, $\boldsymbol{L} = \boldsymbol{U} - \left(\boldsymbol{W}^T + \boldsymbol{W}\right)/2$ is a difference operator known as the *graph Laplacian* and $\boldsymbol{U}$ is a diagonal matrix with its diagonal elements:

$$u_{ii} = \sum_{j=1}^{n} w_{ij}. \tag{2.5}$$

2.2 Extreme Learning Machines

Extreme Learning Machine (ELM) [14, 20] is an emerging popular learning framework for Single-hidden Layer Feedforward neural Networks (SLFNs). ELM has been extended to handle many fundamental machine learning tasks, such as classification [14], regression [14], and clustering [15], and has also been applied in many real-world problems [21, 22, 23].

Figure 2.2 shows an illustration of an SLFN trained with ELM. The architecture of ELM-based SLFNs have two parts: 1) ELM feature mapping and 2) ELM learning. In the ELM feature mapping, given a sample $\boldsymbol{x} \in \mathbb{R}^{n_I}$, the output of the i-th hidden neuron

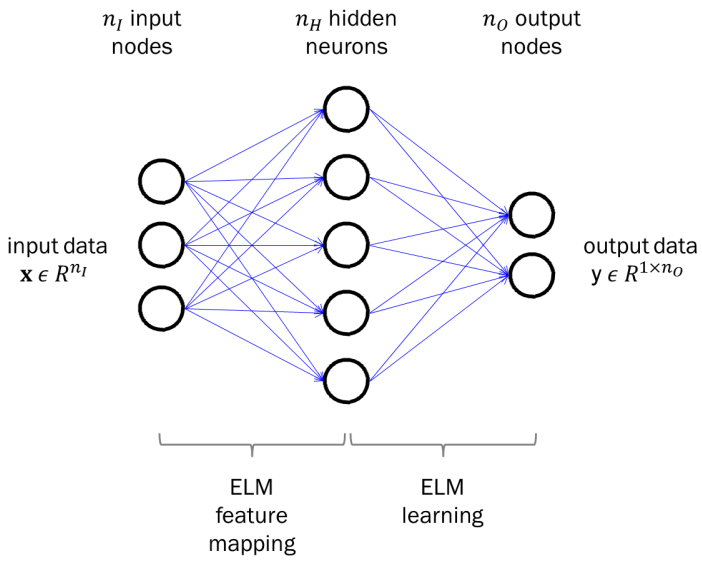


Figure 2.2: Illustration of an SLFN trained with ELM.

with respect to sample $\boldsymbol{x}$ is:

$$h_i(\boldsymbol{x}) = g(\boldsymbol{a}_i, b_i, \boldsymbol{x}), \boldsymbol{a}_i \in \mathbb{R}^{n_I}, b_i \in \mathbb{R}, i = 1, \ldots, n_H, \tag{2.6}$$

where $g(\boldsymbol{a}_i, b_i, \boldsymbol{x})$ is a nonlinear piecewise continuous activation function, n_H is the number of hidden neurons, and $(\boldsymbol{a}_i, b_i)$ are the parameters of the activation functions of the i-th hidden neuron. Some examples of hidden neuron activation functions are as follows:

- Sigmoid function:

$$g(\boldsymbol{a}, b, \boldsymbol{x}) = \frac{1}{1 + \exp\left(-(\boldsymbol{a} \cdot \boldsymbol{x} + b)\right)}; \tag{2.7}$$

- Gaussian function:

$$g(\boldsymbol{a}, b, \boldsymbol{x}) = \exp\left(-b\|\boldsymbol{x} - \boldsymbol{a}\|_2^2\right); \tag{2.8}$$

- Multiquadric function:

$$g(\boldsymbol{a}, b, \boldsymbol{x}) = \left(\|\boldsymbol{x} - \boldsymbol{a}\|_2^2 + b^2\right)^{1/2}; \tag{2.9}$$

- Hard-limit function:

$$g(\boldsymbol{a}, b, \boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{a} \cdot \boldsymbol{x} - b \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

The output vector of the hidden layer is denoted by $\boldsymbol{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_{n_H}(\boldsymbol{x})] \in \mathbb{R}^{1 \times n_H}$, where n_H is the number of hidden neurons. In the ELM learning part, the hidden neurons are connected to n_O output nodes and the weights of the connection are denoted by $\boldsymbol{\beta} \in \mathbb{R}^{n_H \times n_O}$. The final output of ELM with respect to data $\boldsymbol{x}_i$ has the form $\boldsymbol{y}_i = \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} \in \mathbb{R}^{1 \times n_O}$.

The merit of ELM is that the parameters of hidden neurons need not be tuned and can be randomly generated and fixed without sacrificing the universal approximation capability [13], i.e., given any nonconstant piecewise continuous function $g : \mathbb{R}^{n_I} \to \mathbb{R}$, if span$\{g(\boldsymbol{a}, b, \boldsymbol{x}) : (\boldsymbol{a}, b) \in \mathbb{R}^{n_I} \times \mathbb{R}\}$ is dense in ${n_H}^2$, for any continuous target function f and any function sequence $\{g(\boldsymbol{a}_i, b_i, \boldsymbol{x})\}_{i=1}^{n_H}$ randomly generated based on any continuous sampling distribution, $\lim_{n_H \to \infty} |f - f_{n_H}| = 0$ holds with probability one if the output weights $\boldsymbol{\beta}_i$ are determined by ordinary least square to minimize $|f(\boldsymbol{x}) - \sum_{i=1}^{n_H} \boldsymbol{\beta}_i g_i(\boldsymbol{a}_i, b_i, \boldsymbol{x})|$.

Depending on the learning tasks, several variants of ELM have been proposed. A brief review of ELM for supervised classification task is reviewed here. ELM aims to learn a decision rule or approximation function based on the training set $\{\mathcal{X}, \mathcal{T}\} = \{\boldsymbol{x}_i, \boldsymbol{t}_i\}_{i=1}^{n}$, where n is the number of training samples, $\boldsymbol{x}_i \in \mathbb{R}^{n_I}$ is a training sample, and n_I is the number of input neurons. In a regression task, $\boldsymbol{t}_i \in \mathbb{R}^{n_O}$ is the target output of dimension n_O; in a classification task, $\boldsymbol{t}_i \in \{0, 1\}^{n_O}$ is the class label vector with only the k-th entry equal to one, where k is the index of the class that $\boldsymbol{x}_i$ belongs to, and n_O is the number of classes. The output weights can be solved by minimizing the training errors associated with the training data pairs, which leads to the following formulation:

$$\min_{\boldsymbol{\beta}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|_F^2 + \frac{C}{2}\|\boldsymbol{T} - \boldsymbol{H}\boldsymbol{\beta}\|_F^2 \tag{2.11}$$

where $\boldsymbol{H} = [\boldsymbol{h}(\boldsymbol{x}_1)^T, \ldots, \boldsymbol{h}(\boldsymbol{x}_n)^T]^T \in \mathbb{R}^{n \times n_H}$ is the output matrix of the hidden layer, $\boldsymbol{T} = [\boldsymbol{t}_i, \ldots, \boldsymbol{t}_n]^T \in \{0,1\}^{n \times c}$ is the target matrix, and $C > 0$ is a tradeoff parameter. Here the first term in the objective function is a regularization term against over-fitting. This problem can be solved with a closed form solution:

$$\boldsymbol{\beta} = \begin{cases} \left(\boldsymbol{H}^T\boldsymbol{H} + \frac{\boldsymbol{I}}{\lambda}\right)^{-1}\boldsymbol{H}^T\boldsymbol{T}, & \text{if } n_H \leq n, \\ \boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{H}^T + \frac{\boldsymbol{I}}{\lambda}\right)^{-1}\boldsymbol{T}, & \text{otherwise.} \end{cases} \tag{2.12}$$

2.3 Clustering

In machine learning, unsupervised classification or clustering is the task of grouping data points into clusters so that the data points in the same clusters are more similar to each other than those in different clusters [2]. Clustering methods [3] reveal the intrinsic grouping of data and thus have a wide range of applications ranging from engineering, computer sciences, and life sciences to earth sciences, social sciences, and economics [24, 3].

Clustering methods can be broadly divided into two categories: hierarchical and partitional. The hierarchical methods find the nested clusters either in the divisive mode (starting from all data points as one cluster and recursively dividing each cluster into smaller clusters) or agglomerative mode (starting from each data point forming a cluster and recursively merge small clusters into big clusters). On the contrary, the partitional methods find all the clusters simultaneously as a partition of data. As a result, the partitional methods are usually associated with an objective function that measures the "goodness" of a partition and an optimization algorithm that finds the optimal partition.

This thesis focuses on the partitional clustering algorithms. In terms of the input to the algorithm, these partitional algorithms can be broadly categorized into three types: clustering based on data, clustering based on embedding, and clustering based on graph, which will be reviewed in the following sections.

2.3.1 Clustering on Data

One of the earliest and most popular partitional algorithms, k-means, traced back to 1950s [2]. Given a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times n_I}$, where $\boldsymbol{x}_i \in \mathbb{R}^{n_I}$ denotes the i-th sample, k-means aims to find a partition such that the squared error the data points in the cluster and their corresponding cluster centroids is minimized. The objective function is as follows.

$$\min_{\mathcal{X}_i, \boldsymbol{g}_i, i=1,\ldots,c} \quad \sum_{i=1}^{c} \sum_{\boldsymbol{x}_j \in \mathcal{X}_i} (\boldsymbol{x}_j - \boldsymbol{g}_i)^T (\boldsymbol{x}_j - \boldsymbol{g}_i) \tag{2.13}$$

where $\boldsymbol{g}_i \in \mathbb{R}^d$ is the centroid of the i-th cluster, and $\mathcal{X}_i$ is the set of data points belonging to the i-th cluster.

Let $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n]^T \in \{0,1\}^{n \times c}$ be the cluster indicators, where $\boldsymbol{f}_i = [f_{i1}, f_{i2}, \ldots, f_{ic}]^T \in \{0,1\}^c$, $f_{ij} = 1$ if and only if $\boldsymbol{x}_i$ belongs to the j-th cluster and $f_{ij} = 0$ otherwise, and let $\boldsymbol{G} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_c] \in \mathbb{R}^{n_I \times c}$ be the cluster centroid matrix. The objective function of k-means can be transformed into a matrix form as

$$\begin{aligned}
&\sum_{i=1}^{c} \sum_{\boldsymbol{x}_j \in \mathcal{X}_i} \left(\boldsymbol{x}_j - \boldsymbol{g}_i\right)^T \left(\boldsymbol{x}_j - \boldsymbol{g}_i\right) \\
&= \sum_{j=1}^{n} \left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right)^T \left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right) \\
&= \sum_{j=1}^{n} \mathrm{Tr}\left(\left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right) \left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right)^T\right) \\
&= \mathrm{Tr}\left(\sum_{j=1}^{n} \left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right) \left(\boldsymbol{x}_j - \boldsymbol{G}\boldsymbol{f}_j^T\right)^T\right) \\
&= \mathrm{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{F}\boldsymbol{G}^T\right) \left(\boldsymbol{X} - \boldsymbol{F}\boldsymbol{G}^T\right)^T\right) \\
&= \|\boldsymbol{X} - \boldsymbol{F}\boldsymbol{G}^T\|_F^2.
\end{aligned} \tag{2.14}$$

To solve this problem, the main steps of k-means are: 1) generate an initial partition with k clusters (e.g., randomly); 2) compute the cluster centroids; 3) generate a new

partition by assigning each data point to its nearest cluster centroid; 4) go to step 2, until convergence.

In k-means, the clustering criterion is the squared error between data points and its nearest cluster centroid. The other commonly used clustering criteria are defined based on the concepts of with-cluster scatter matrix $\boldsymbol{S}_w$, between-cluster scatter matrix $\boldsymbol{S}_b$, and total scatter matrix $\boldsymbol{S}_t$. Give a set of data points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, the three matrices associated with a partition $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_c\}$ are defined as follow:

$$\begin{aligned}
\boldsymbol{S}_w &= \sum_{j=1}^{c} \sum_{\boldsymbol{x} \in \mathcal{X}_j} \big(\boldsymbol{x} - \boldsymbol{g}_j\big)\big(\boldsymbol{x} - \boldsymbol{g}_j\big)^T, \\
\boldsymbol{S}_b &= \sum_{j=1}^{c} n_j \big(\boldsymbol{g}_j - \bar{\boldsymbol{x}}\big)\big(\boldsymbol{g}_j - \bar{\boldsymbol{x}}\big)^T, \\
\boldsymbol{S}_t &= \sum_{i=1}^{n} \big(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\big)\big(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\big)^T,
\end{aligned} \tag{2.15}$$

where c is the number of classes, $\mathcal{X}_j$ is the set of samples of the j-th class, n_j is the size of $\mathcal{X}_j$, $\boldsymbol{g}_j$ is the centroid of j-th class, and $\bar{\boldsymbol{x}}$ is the global centroid. Following the definition, the equation $\boldsymbol{S}_t = \boldsymbol{S}_b + \boldsymbol{S}_w$ holds. Furthermore, $\boldsymbol{S}_t$ can also be calculated using the data matrix as $\boldsymbol{S}_t = \bar{\boldsymbol{X}}^T \bar{\boldsymbol{X}}$, where $\bar{\boldsymbol{X}} = \big(\boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\big)\boldsymbol{X}$ is the mean centred data matrix, and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times n_I}$.

Based on these concepts, three popular clustering criteria [4] are 1) minimization of $\mathrm{Tr}\,(\boldsymbol{S}_w)$ and 2) minimization of $\det(\boldsymbol{S}_w)$, 3) maximization of $\mathrm{Tr}\,(\boldsymbol{S}_b(\boldsymbol{S}_w)^{-1})$. Various algorithms are developed to solve for the optimal partitions of the data that satisfy these criteria.

2.3.2 Clustering on Embedding

Data in the original space may contain noise and redundant features. One way to alleviate this problem for better clustering performance is to conduct clustering using an improved representation of the data, called an embedding [25]. Traditional embedding

approaches [26] aim to find a subspace where data are presented with maximum variance, as in Principal Component Analysis (PCA), or maximum separability, as in Linear Discriminant Analysis (LDA). Motivated by the complex structures of data, researchers proposed many nonlinear feature learning methods, also called embedding methods, such as locality linear embedding [27], Laplacian eigenmap [19], and isometric mapping [28]. These embedding methods learn features by preserving important local or global structures of the data while discarding the redundant or noisy information.

Embedding and clustering are usually conducted in sequence to achieve better clustering results, since cluster structures may not be prominent in the original feature space. For instance, PCA can be used for dimensionality reduction followed by k-means for clustering [29, 30].

Alternatively, embedding and clustering can be conducted joinly. The advantage of joint embedding and clustering is that, with the clustering solution available as label information, supervised methods, such as LDA, can also be used for embedding [29]. Intuitively, better clustering performance is expected from the joint embedding and clustering methods because the requirement of clustering is taken into account during the embedding step. As a well-known example, Ding and Li [29] proposed to conduct LDA and k-means alternately, which yields significantly better clustering performance compared with the counterpart of conducting PCA and k-means in sequence. This method is later proved to be equivalent to kernel k-means with a specific kernel function [31]. Recently, Hou et al. [11] proposed a discriminative embedded clustering (DEC) framework that unifies several joint embedding and clustering techniques. In these methods, embedding is achieved by linear transformation, which has limited capability in discovering the nonlinear structures of the data. Hence, it is desirable to look for methods which can perform nonlinear transformation for embedding.

The SLFNs trained using ELM can be used as effective nonlinear transformation tools, thanks to ELM's universal approximation capability [13]. If more clear and accurate cluster structures can be observed afterward, such transformation is considered benefi-

cial for clustering. Motivated by this fact, several research studies have been carried out on finding suitable criteria for ELM to achieve the optimal nonlinear transformation. He et al. [32] proposed to conduct Clustering in ELM Feature Space (C-ELM-FS) to achieve nonlinear embedding, but C-ELM-FS omits the output weights and the output layer and thus sacrifices the flexibility of ELM. The other ELM-based embedding for clustering methods make use of the complete SLFNs and can be broadly divided into two groups.

The first group aims to capture the intrinsic structure of the data in the original space. Huang et al. [15] extended ELM to embedding and clustering under the manifold regularization framework and called the method Un-Supervised ELM (US-ELM). The main idea is to learn a nonlinear data embedding which preserves the intrinsic manifold structure and subsequently conduct clustering in the embedded space. Peng et al. [33] proposed Unsupervised Discriminative ELM (UD-ELM), which preserves both local manifold structure and global discriminative information in the original data space. However, UD-ELM requires the dimension of embedded space to be equal to the number of classes, which sacrifices the flexibility of the data representation.

The second group uses ELM to represent the data in a way such that clusters are better revealed in the embedded space. Liyanaarachchi et al. [34] propose to conduct clustering in the embedded space whose projection matrix is solved by ELM Auto-Encoder. The study shows that such embedding can effectively increases the class separability by preserving the between-class variance and reducing the within-class variance in the embedded space. This method is referred to as clustering based on ELM-Auto-Encoder (C-ELM-AE). Huang et al. [35] proposed three ELM-based Discriminative Clustering (ELM-DC) methods. The motivation is to solve ELM embedding and cluster indicators iteratively such that the embedding could lead to good classification results as measured by the objective functions of supervised methods, i.e., LDA or ELM. A better classification result implies higher class separability of the data in the embedded space.

In the following sections, two related methods are reviewed in details. US-ELM is a

state-of-the-art nonlinear ELM-based embedding method for clustering. DEC is a state-of-the-art linear method, which conducts embedding and clustering jointly. Given the unlabeled training data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times n_I}$ and the desired number of clusters c, the goal of embedding-based clustering is to obtain the embedding and the cluster indicator of all data points.

Unsupervised Extreme Learning Machine

UnSupervised Extreme Learning Machine (US-ELM) [15] aims to learn a nonlinear mapping which preserves the underlying structure of the original data based on manifold regularization framework [36]. The output of ELM, i.e., $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{\beta} \in \mathbb{R}^{n \times n_O}$, is the nonlinear embedding of the data and the cluster indicators are obtained by conducting off-the-shelf clustering algorithms using the embedding.

To obtain the nonlinear embedding, specifically, US-ELM algorithm first constructs a similarity matrix $\boldsymbol{W}$ based on the data in the original feature space and then calculates the graph Laplacian $\boldsymbol{L}$. Then the output weights $\boldsymbol{\beta}$ are solved by preserving the similarity relationship between data points, which leads to the following formulation:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{\beta}\|_F^2 + \lambda \operatorname{Tr}\left(\boldsymbol{\beta}^T \boldsymbol{H}^T \boldsymbol{L} \boldsymbol{H} \boldsymbol{\beta}\right) \\ \text{s.t.} \quad & (\boldsymbol{H}\boldsymbol{\beta})^T \boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{I}_{n_O} \end{aligned} \tag{2.16}$$

The first term in the objective function is a regularization term against overfitting, the second term based on manifold regularization preserves the local structure of the data, λ controls the trade-off between the two terms, and the constraint is added to avoid trivial solution of $\boldsymbol{\beta}$ where all elements are zero. The optimal solution can be obtained by solving a generalized eigenvalue problem.

Discriminative Embedded Clustering

Discriminative Embedded Clustering (DEC) [11] is a recently proposed framework which unifies several existing methods for joint embedding and clustering. It explicitly conducts embedding and clustering in the same formulation and uses an iterative algorithm to find the local optimal solution. From the perspective of embedding, DEC essentially finds a linear transformation matrix $\boldsymbol{P} \in \mathbb{R}^{n_I \times n_O}$ that maximizes the class separability [37], so that the linear embedding is obtained as $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{P} \in \mathbb{R}^{n \times n_O}$. In the same formulation, DEC solves the cluster indicator matrix $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n]^T \in \mathbb{R}^{n \times c}$ and the cluster centroid matrix $\boldsymbol{G} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_c] \in \mathbb{R}^{n_O \times c}$, where $\boldsymbol{g}_i \in \mathbb{R}^{n_O}$ is the centroid of i-th cluster and $\boldsymbol{f}_i = [f_{i1}, f_{i2}, \ldots f_{ic}]^T \in \mathbb{R}^c$, $f_{ij} = 1$ if and only if $\boldsymbol{x}_i$ is assigned to the j-th cluster and $f_{ij} = 0$ otherwise. Mathematically, the formulation of DEC is as follows:

$$\begin{aligned} \max_{\boldsymbol{P},\boldsymbol{G},\boldsymbol{F}} \quad & \operatorname{Tr}\left(\boldsymbol{P}^T \boldsymbol{S}_t \boldsymbol{P}\right) - \lambda \left\| \boldsymbol{X}\boldsymbol{P} - \boldsymbol{F}\boldsymbol{G}^T \right\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{P}^T \boldsymbol{P} = \boldsymbol{I}_{n_O} \end{aligned} \tag{2.17}$$

As can be seen from the above formulation, the first term in the objective function aims to maximize the total variance of data points in the embedded space, similar to the objective of PCA. The second term is derived from the objective of k-means, i.e., minimizing the squared sum of distances of data points to their respective cluster centroids. It has been proved [11] that when $\lambda > 1$, the formulation is equivalent to maximizing the between-class separation and minimizing the within-class variance; when $\lambda = 1$, the formulation is equivalent to only maximizing the between-class separation; when $0 < \lambda < 1$, both the between-class separation and within-class variances are maximized, but DEC puts a larger weight on the between-class separation.

Since the problem is not a jointly convex problem, Hou et al. [11] proposed to solve this problem iteratively: 1) fixing $\boldsymbol{P}$, $\boldsymbol{G}$ and optimizing $\boldsymbol{F}$, which is similar to cluster assignment process in k-mean given fixed cluster centroids; 2) fixing $\boldsymbol{F}$ and optimizing

P, G. Then the two steps are iteratively performed.

2.3.3 Clustering on Graph

Rather than taking the data as the input, many effective clustering methods take their similarity, in the form of a graph, as the input. As a popular graph-based clustering method, spectral clustering was first proposed in the 1970s and has become increasingly popular in the recent decade [38]. The main procedure involves constructing a similarity matrix, calculating the eigenvectors of the graph Laplacian as a low dimension data representation, and obtaining cluster indicators based using some off-the-shelf algorithms. Unlike many traditional clustering methods which assume convex clusters, spectral clustering methods do not make strong assumptions on the shape of the clusters, because they take the connection among data points as the input.

Many research studies investigate how to construct better similarity matrix: The commonly used methods are the ϵ neighborhood graph, k nearest neighbor graph, and fully connected graph, which can be constructed based on traditional distances (e.g. Euclidean distance) or some advanced hybrid similarities [39]. Hou et al. [40] proposed to standardize the graph similarity matrices by histogram equalization before clustering. Zelnik-Manor and Perona [41] proposed to optimize the graph construction parameters based on data locally. However, these traditional graph construction methods are all optimized independently of the subsequent tasks.

Recently, many researchers adopt the idea of learning the graph jointly with the subsequent tasks, such as feature selection [42], dimensionality reduction [43], and image segmentation [44], so that the requirement of the subsequent tasks are taken into account during the graph learning. Such adaptive graph learning methods have shown promising performance. Nie et al. proposed several clustering methods, namely Clustering with Adaptive Neighbors (CAN) [16], Projected Clustering with Neighbors (PCAN) [16], and Constrained Laplacian Rank (CLR) [17], to learn the similarity matrix and cluster-

ing structure simultaneously. The common feature of these clustering methods is the imposed constraint on the graph Laplacian. The difference among these three methods mainly lies in the data representation for graph learning.

The details of these adaptive graph learning methods are explained in the following subsections. Given a set of unlabeled training data $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^{n_I}$, and the target number of clusters c, the adaptive graph learning methods aim to learn a data similarity matrix $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n] = [w_{ij}] \in \mathbb{R}^{n \times n}$ and the corresponding graph Laplacian of the learned graph is denoted as $\boldsymbol{L}$ calculated as in Section 2.1.

Clustering with Adaptive Neighbors

Clustering with Adaptive Neighbors (CAN) enforces the graph's consistency with the data structure by learning the similarity matrix of the graph in the following way: a small distance between a pair of data points in the original feature space should correspond to a large similarity value, which can be mathematically formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 w_{ij} + \gamma w_{ij}^2 \right) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1, \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{2.18}$$

Projected Clustering with Adaptive Neighbors

Extending CAN to handle high dimensional data, Projected Clustering with Adaptive Neighbors (PCAN) learns the similarity matrix of the graph based on the pairwise distances in a linear subspace and solves the transformation matrix $\boldsymbol{P} \in \mathbb{R}^{n_I \times n_O}$ jointly,

where n_O is the dimensionality of the linear subspace.

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{P}} \quad & \sum_{i,j=1}^{n} \left(\left\| \boldsymbol{P}^T \boldsymbol{x}_i - \boldsymbol{P}^T \boldsymbol{x}_j \right\|_2^2 w_{ij} + \gamma w_{ij}^2 \right) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1, \\ & \boldsymbol{P}^T \boldsymbol{S}_t \boldsymbol{P} = \boldsymbol{I}_{n_O}, \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{2.19}$$

Constrained Laplacian Rank

Different from learning a graph based on the training data, Constrained Laplacian Rank (CLR) learns a new graph based on a given graph $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. The initial graph $\boldsymbol{A}$ is calculated based on the training data in the original feature space (referring [17] for details). The underlying assumption is that this initial graph retains the similarity relationship between data points. As such, the difference, e.g., measured using the Frobenius norm and L1-norm, between the given graph and the new graph should be as small as possible. Based on the two norms, two formulations for graph learning are as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \|\boldsymbol{W} - \boldsymbol{A}\|_F^2 \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1, \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{2.20}$$

and

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \|\boldsymbol{W} - \boldsymbol{A}\|_1 \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1, \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{2.21}$$

2.4 Semi-Supervised Classification

In the recent years, Semi-Supervised Classification (SSC) has become an active machine learning topic and has attracted significant attention from various research communities [5, 45]. SSC algorithms construct the classification model using a small amount of training data with labels and a large amount of training data without labels. They satisfy the need of many real world applications where collecting labels are either costly or time-consuming but unlabeled training samples are plentiful.

One example of early SSC algorithms is self training [45]: a model is first constructed using only labeled data; after predictions are finished, unlabeled data with high confidence are converted into labeled data according to the predictions; then further training is conducted based on the updated labeled data; these three processes are then repeated iteratively. Self training is a general wrapper that can be applied to let many supervised learning algorithms make use of unlabeled data. Over the past decade, two types of advanced SSC algorithms have emerged. In addition to minimizing the empirical training error, both types further regularize the decision boundary based on feature vectors of training data according to some assumptions:

- **Low-density-separation-based methods.** The decision boundary is forced to move towards the low-density region of the training data. The underlying assumption is that data from the same class lie in the same cluster, which is also known as the cluster assumption. The example algorithms are Transductive Support Vector Machine (TSVM) [46] and Semi-Supervised Support Vector Machine (S^3VM) [47].

- **Graph-based methods.** In many applications, high-dimensional data lie on a low dimensional manifold. This is known as the manifold assumption. The data manifold consists of many local structures represented using a undirected weighted graph. In graph-based methods, the decision boundary is forced to be smooth with respect to data manifold. The decision boundary learned in this way pre-

serves the local structures of the data manifold. In other words, samples that are close to each other will be predicted with the similar class labels. Popular graph-based methods include Laplacian Support Vector Machine (LapSVM) [36] and Semi-supervised Extreme Learning Machine (SS-ELM) [15].

2.4.1 Manifold Regularization

As shown in Figure 2.3, the data points exist in a three-dimensional space. However, instead of spreading over the complete three-dimensional space, they tend to reside on a 'swiss roll' shape structure. If the surface of the 'swiss roll' is viewed as a space, then each data point can be specified using two dimensions. In other words, this set of three-dimensional data lies on a two-dimensional manifold. In fact, many real-world applications generate high-dimensional data that lie on low-dimensional manifolds [28, 48].

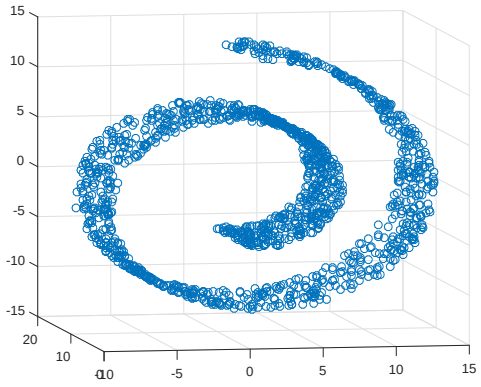


Figure 2.3: Illustration of the "swiss roll" data.

Based on this assumption, Belkin et al. [36] proposed manifold regularization framework, which has drawn considerable attentions from research by far. In brief, a n-dimensional manifold consists of many small local neighborhoods, where data points homomorphic to the Euclidean space of dimension n. To force the decision boundary to be 'smooth' with respect to the manifold, the formulation penalizes large difference in the predicted output with respect to two nearby data points. Here the similarity matrix

$\boldsymbol{W} = [w_{ij}]$ is computed as in Section 2.1. Therefore, the mathematical formulation is as follows:

$$L_m = \frac{1}{2} \sum_{i,j} w_{ij} |y_i - y_j|^2, \tag{2.22}$$

where y_i and y_j are the predicted output with respect to two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. This problem is usually represented using its equivalent matrix form:

$$L_m = \boldsymbol{y}^T \boldsymbol{L} \boldsymbol{y}, \tag{2.23}$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$ is the output vector, and $\boldsymbol{L} = \boldsymbol{U} - \left(\boldsymbol{W}^T + \boldsymbol{W}\right)/2$ is a difference operator known as the graph Laplacian and $\boldsymbol{U}$ is a diagonal matrix with its diagonal elements:

$$u_{ii} = \sum_{j=1}^{n} w_{ij}. \tag{2.24}$$

2.5 Driver Distraction Detection

Extensive research has been done on developing driver distraction detection systems. As the primary step, diagnostic measures, such as eye and head movements, need to be collected from a driver during driving either in a simulated environment or on the road. The construction of driver distraction detection system can be broken down into two major works: 1) labeling the data according to the distraction state and 2) developing a detection model using the data.

2.5.1 Eye and Head Movements

The collection of eye and head movements is done using normal cameras or advanced commercial eye trackers. Normal cameras output only image sequences of the driver's

face. The normal camera-based driver detection systems include computer vision techniques as the first step so as to obtain the rough eye and head movements [49]. Advanced commercial eye trackers are built with robust cameras and image processing modules. They can directly provide information about subtle eye movements with finer time resolutions and higher tracking accuracy, which is beneficial for detecting cognitive distraction. This section focuses on eye and head movements collected from commercial eye trackers. Figure 2.4 shows different levels of eye movement features.

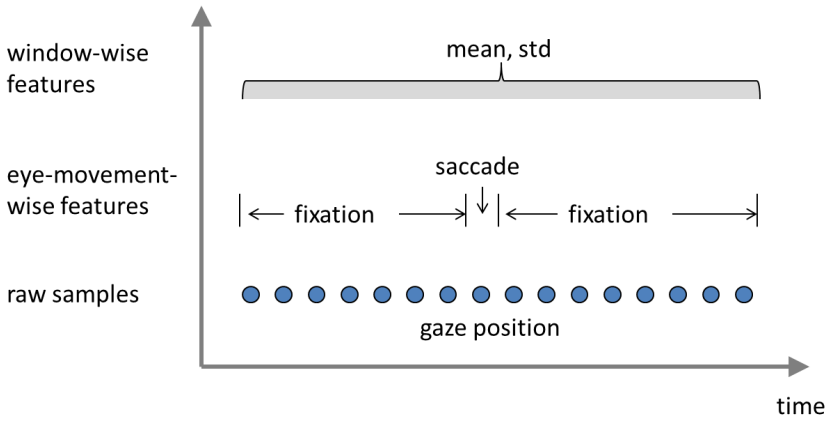


Figure 2.4: Illustration of different levels of eye movement features.

Raw samples consist of several measures, including the gaze rotation (/position), head orientation, pupil diameter, eyelid opening, and active area of interest. They are generated at a sampling frequency between 30 Hz to 60 Hz depending on the eye tracker. The gaze positions are in nature time series and form some consistent temporal patterns. Particularly, three temporal patterns have been found by psychological research to effectively reflect humans' ways of capturing visual information:

- A fixation consists of a temporally consecutive gazes that fall within a small range and happens when the person is extracting information from a stationary object (an example is shown in Figure 2.5);
- A saccade consists of a temporally consecutive gazes that move largely within a short period and happens when the person is redirecting their visual attention from one object to another (an example is shown in Figure 2.5).

- A smooth pursuit usually happens when the person is extracting information from a slowly moving object and is characterized by gaze movements that are slower than those in a saccade but move within a larger range than those a fixation.

In addition, a blink is defined as a rapid eye closure followed quickly by a rapid eye opening and is another commonly observed eye behavior.

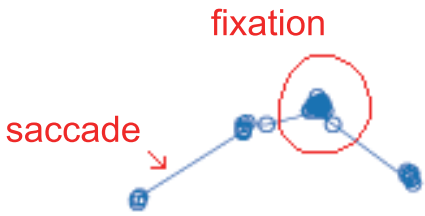


Figure 2.5: Example plot of gaze positions.

Table 2.1 shows a summary of eye-movement-wise features from the literature. The type of eye movements is first identified from raw samples segments, and then features of these eye movements can be extracted.

Table 2.1: A summary of the eye-movement-wise features

Eye Movement	Features	Example
fixation	position, duration	[8]
saccade	velocity, duration, peak velocity over duration	[50]
smooth pursuit	distance, direction, velocity, duration	[8]
blink	frequency, duration	[8, 9]

Distraction is the diversion of attention away from activities critical for safe driving toward competing activities [51]. In this thesis, the state is regarded as “distracted” if the driver is engaged in non-driving-related activities or “not distracted” if the driver is fully focused on driving. The distraction state of a person usually cannot be revealed from a single eye movement at the moment. Therefore, it is important to capture the

temporal contextual information of many eye movements. Statistical functionals, such as mean and standard deviation, in sliding windows are commonly used to capture the temporal contextual information.

2.5.2 Labeling

In the naturalistic driving scenario, distraction can happen at any time depending on the driver, e.g., thinking about the meeting to be held at his destination. In contrast, in an experiment scenario the driver is instructed to perform some secondary tasks to simulate driver distraction. In either case, researchers need to create labels, i.e., the true distraction states, for each sample throughout the data collection period. From the literature, four popular methods to generate labels have been found:

- Driving performance. This method is also called surrogate distraction measurement [51]: degradation in driving performance is considered as the indicator of distraction given that other factors are held unchanged. Generally, this labeling method requires setting up a test driving scenario where driving task performance can be measured. For example, in the lane change test (LCT), the participant need to drive and perform lane change according to signs along the road [51]; degradation of the LCT performance is measured and used as the reference for labeling.

- Secondary task. If the distraction is stimulated by secondary tasks given by the experimenters, labeling can be done based on the presence of the secondary task or the performance of the secondary task. For example, the complete recording of the sections where the driver performs the secondary task is labeled as "distracted", whereas the control sections without secondary tasks are labeled as "not distracted" [8, 10, 52, 53, 54, 55, 56, 57].

- Subjective assessments. In this method, the assessments are carried out by either the drivers themselves based on their feelings or other external evaluators based on their observations of the driver, e.g., from a video recording [9]. In either case,

some evaluation standards (e.g. NASA task load index) are provided to make sure the assessment process is standardized [58].

- Other diagnostic measures. For example, in the study conducted by Tango and Botta [59], vehicle-based measures were used to detect distraction, while gaze positions were used to generate the labels. In another study [8], where eye movements were used to detection distraction, vehicle-based measures, such as steering error values and accelerator release reaction times, were used as the labeling reference. Specifically, the upper quartile of the two measures for each participant was considered as "distracted" and the remainder as "not distracted."

Among the four labeling methods, the secondary task method is a very common way of labeling, because it is very cost efficient if the timing of secondary task is controlled and recorded. However, this method may not generate accurate labels in situations where the driver did not follow the instructions on secondary task exactly. The subjective assessment method by external evaluators can effectively overcome this drawback because each sample is labeled separately and directly. Nonetheless, it is worth noting that the subjective assessment method by external evaluators requires extensive work, e.g., the development of a labeling software with GUI and the training of external evaluators. Moreover, as reported by a recent study [9], the actual labeling time for 480 video segments (10 seconds each) goes up to approximately 21.5 man-hours. In naturalistic driving, where the driver constantly voluntarily change his state, the labeling work can be more extensive. In addition, some researchers also combine several ways of labeling according to various needs. For example, in one study [53] the secondary task was used to generate labels of "distracted" and "not distracted," and the subjective assessment by the driver were used to generate labels of degrees of distraction. Subjective assessment by external evaluators usually requires other data source, e.g., face video recording.

2.5.3 Modeling

The development of distraction detection models can be classified into two categories:

Rule-based Approach. Some patterns can be discovered via analyzing the diagnostic measures and can be then formed into rules for detecting distraction. For example, a high blink rate and a shrink in the visual searching field were observed in driving with auditory-mental (cognitive) secondary tasks [54]. Similarly, Nakayama et al. [52] developed a steering entropy based method for distraction detection. The underlying assumption is that a driver's steering behavior tends to become discontinuous while the driver performs an activity in addition to driving. They quantified these discontinuities using steering entropy and proposed to evaluate driver workload based on steering entropy. Such approaches improve the understanding of human cognition process and driving behavior but ignore the diagnostic effects of combining many measures, since each measure is studied independently. Moreover, due to the individual difference, very few measures can be found to have high indicative power for a wide range of drivers.

Machine-learning-based Approach. Recently, it has been a trend to use machine learning techniques to develop detection models. This approach is less restrained by the incomplete knowledge of the relationship between diagnostic measures and humans' cognitive process, because it can explore multiple diagnotistic measures jointly. Zhang et al. [10] were among the pioneers to use machine learning approaches to detect driver cognitive distraction. In their work, decision tree was adopted as the learning algorithm. Kutila et al. [57] used Support Vector Machines (SVMs) for driver cognitive distraction detection in real driving conditions. Liang et al. used eye and head movements collected by eye tracker as the main input for driver cognitive distraction detection. SVM [8], Bayesian networks [55], and a hybrid Bayesian network [56] have been used. Tango and Botta [59] compared several machine learning methods for visual distraction detection. In their work, SVM achieved the best detection performance. Wölmer et al. [53] investigated long short term memory neural networks. Because of their ability to capture contextual information, they achieve an outstanding visual distraction detection

accuracy of up to 96.6%. Li et al. [9] modeled the distraction detection problem as regression and classification problems. Six classifiers were evaluated, and some promising results were obtained for subject-independent models.

In terms of learning and generalization capability, the evolution of machine learning based driver distraction systems follows three stages:

• Driver-specific model. The model is developed using training data from one specific driver, who will eventually be the user.

• Driver-independent model. The model is developed using training data from many drivers and is expected to serve new drivers that have not been seen.

• Driver-independent model that adapts to new drivers. The model adapts to a new driver by updating itself using data collected as the driver is using the system.

The state-of-the-art research is still focusing on the first stage [56, 59], and has demonstrated that it is promising to detect driver distraction in real time based on various diagnostic measures. Moving from the first to the second stage requires a training data from a large number of drivers, which is usually beyond the capability of an academic research setting. This thesis focuses on the development of the driver-specific model.

Chapter 3

Joint Embedding and Clustering via Extreme Learning Machine

In order to improve the clustering performance, Chapter 3 explores the embedding-based clustering methods, with a focus on the methods that employ ELM as an efficient embedding tool. By incorporating all the merits of existing methods presented in Section 3.1, Section 3.2 proposes a method named ELM for Joint Embedding and Clustering (ELM-JEC). The proposed algorithm is tested on the benchmark datasets in Section 3.3.

3.1 Background and Motivation

Extreme Learning Machine (ELM) is an efficient and flexible embedding tool which has the potential to realize different properties of embedding. As reviewed in Section 2.3.2, the ELM-based clustering methods can be broadly divided into two group: 1) using ELM to capture the intrinsic structure of the data in the original space before conducting clustering in the embedded space; 2) using ELM to represent data such that clusters are better revealed before conducting clustering in the embedded space. While both groups achieve the desirable property of nonlinearity efficiently, they have some drawbacks. The first group learns the embedding without considering the requirement of clustering, as they conduct embedding and clustering separately. This could result in overlapping cluster structures or even no cluster structures in the embedded space, which may affect the clustering performance. The second group tries to obtain an embedded space with clearer cluster structures but ignores the intrinsic structure of the data in the original space. The embedding obtained in this way may alter the local structure of the data and may result in artificial clusters in the embedded space, which could lead to wrong clustering results.

In order to analyze the advantage and limitations of existing work, Table 3.1 summarizes the properties of the embedding-based clustering methods. Three properties associated with embedding and a property associated with clustering are explained in the following:

Nonlinearity. If data in the embedded space cannot be written as a linear combination of the data in the original space, the embedding is regarded as nonlinear. Nonlinearity is a desired property of embedding methods, as data from real-world applications usually form clusters that are not linearly separable and cannot be revealed by linear embedding methods, such as Principle Componenet Analysis (PCA) [28]. For example, Clustering in ELM Feature Space (C-ELM-FS) [32], Unsupervised ELM (US-ELM) [15], Unsupervised Discriminative ELM (UD-ELM) [33], Clustering with ELM Auto-Encoder (C-ELM-AE) [34], and ELM-based Discriminative Clustering (ELM-DC) [35] are all nonlinear in nature as they adopt ELM to approximate the embedding.

Table 3.1: Desirable properties for embedding and clustering

	Property	C-ELM-FS	US-ELM	UD-ELM	C-ELM-AE	LDA-KM	ELM-DC	DEC	**ELM-JEC**
Embedding	Nonlinearity	✓	✓	✓	✓	✗	✓	✗	✓
	Structure Preserving	✗	✓	✓	✗	✗	✗	✗	✓
	Separability Maximizing	✗	✗	✗	✓	✓	✓	✓	✓
Clustering	Simultaneous embedding and clustering	✗	✗	✗	✗	✓	✓	✓	✓

A "✓" symbol indicates the method (on the same column as the symbol) has the property (on the same row as the symbol), while "✗" indicates the absence of the property.

Structure Preserving. The second property is preserving the intrinsic geometric structure of the data in the original space. High-dimensional data usually reside on low-dimensional manifolds that are essential for representing the data effectively. Therefore, the desirable embedded data should have the same intrinsic geometric structure as data in the original space [25]. For example, US-ELM [15] adopts the manifold regularization to achieve maximum smoothness of the embedding with respect to the manifold structure of the data. In UD-ELM [33], both local manifold structure and the global discriminant structure are captured by the embedding, while the redundant or noisy information is discarded.

Separability Maximizing. The third property is maximizing the class separability of the data in the embedded space. Since the purpose of embedding is to improve the performance of clustering, data in the embedded space should have more prominent clusters structure. In other words, the embedding should maximize the class separability [37] of the data in the embedded space. For example, C-ELM-AE achieves this by utilizing the embedding function solved by ELM-AE, which preserves the between-class variance and reduces the within-class variance. Adaptive LDA-guided k-means clustering (LDA-KM) [29] directly adopts LDA and alternates between LDA and k-means. ELM-DC achieves maximum class separability by using supervised embedding methods, i.e., LDA and ELM. The objective function of ELM can be interpreted as minimizing the within-class variance, where the class centroids are fixed as the unit vectors in the c-dimensional embedded space (c is the number of clusters). DEC [11] maximizes the class separability of the data in the embedded space by combining the objective functions of PCA and k-means.

Simultaneous Embedding and Clustering. Simultaneous embedding and clustering are achieved if both the embedding and the clustering results can be obtained using the same approach. The main disadvantage of conducting embedding and clustering in sequence is that other off-the-shelf clustering methods may introduce additional uncertainty and the embedding may not be optimized for a specific clustering method. Only LDA-KM, ELM-DC, and DEC can jointly conduct embedding and clustering, whereas the

other methods in Table 3.1 rely on separate off-the-shelf clustering methods. However, LDA-KM and DEC use a linear transformation for embedding. Although ELM-DC can approximate nonlinear feature mappings, it focuses only on the cluster structure in the embedded space and ignores the intrinsic structure of the data in the original space.

As can be seen from Table 3.1, no existing methods simultaneously satisfy all the four properties mentioned above. To close the research gap, this chapter proposes a method that satisfies all desirable properties specifically by 1) preserving the manifold structure of the data in the original space and 2) maximizing the class separability of the data in the nonlinear embedded space. In brief, the nonlinear embedding is approximated by an ELM-based SLFN, whose the output weights are solved together with cluster indicator variables. In the optimization objective function, the first property is formulated based on the manifold regularization [36], and the second property is formulated based on DEC [11].

3.2 Proposed Method

In a clustering task, given a desired number of clusters c and a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times n_I}$, the goal is to find the cluster indicator matrix $\boldsymbol{F}$ and the cluster centroid matrix $\boldsymbol{G}$. Here $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n]^T \in \{0,1\}^{n \times c}$ and $\boldsymbol{f}_i = [f_{i1}, f_{i2}, \ldots f_{ic}]^T$, where $f_{ij} = 1$ if and only if $\boldsymbol{x}_i$ is assigned to the j-th cluster and $f_{ij} = 0$ otherwise; $\boldsymbol{G} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_c] \in \mathbb{R}^{n_O \times c}$, where $\boldsymbol{g}_i \in \mathbb{R}^{n_o}$ is the centroid of the i-th cluster.

The main idea of ELM for Joint Embedding and Clustering (ELM-JEC) is to jointly approximate a nonlinear embedding using ELM and solve for the clusters based on the data in the embedded space. The desired embedding can be obtained by adjusting only the output weights between the hidden layer and the output layer, because, according to ELM theory, the parameters of hidden layers can be randomly generated and fixed. In order to satisfy the properties of both *structure preserving* and *separability maximizing*, the following objective function is maximized with respect to the output weight $\boldsymbol{\beta} \in$

$\mathbb{R}^{n_H \times n_O}$, the cluster centroid matrix $\boldsymbol{G}$, and the cluster indicator matrix $\boldsymbol{F}$:

$$\begin{aligned} \max_{\boldsymbol{\beta},\boldsymbol{G},\boldsymbol{F}} \quad & \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) - \lambda \left\| \bar{\boldsymbol{H}}\boldsymbol{\beta} - \boldsymbol{F}\boldsymbol{G}^T \right\|_F^2 - \gamma \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{L} \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) \\ \text{s.t.} \quad & \boldsymbol{\beta}^T \boldsymbol{\beta} = \boldsymbol{I}_{n_O} \end{aligned} \tag{3.1}$$

where $\boldsymbol{L}$ is the *graph Laplacian* as defined in Section 2.4.1, $\boldsymbol{I}_{n_O}$ is the identity matrix of n_O dimensions, and $\lambda > 0$ and $\gamma > 0$ are the user-defined trade-off parameters controlling impacts of minimizing within-class variance and preserving manifold structure, respectively. The hidden layer output matrix are transformed to have zero mean. The centered output matrix from the hidden layer is denoted by $\bar{\boldsymbol{H}} = (\boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_n \mathbf{1}_n^T)\boldsymbol{H}$, and $\bar{\boldsymbol{H}}$ is the output matrix of ELM hidden neuron calculated based on $\boldsymbol{X}$ as explained in Section 2.2. The output of the neural network, given as $\bar{\boldsymbol{H}}\boldsymbol{\beta}$, is the embedding of the data in the output space.

The first term in the objective function maximizes the total variance of data in the ELM output space. The second term minimizes the squared sum of distances of data points to their respective cluster centroids in the ELM output space. Together, the first and second terms aim to maximize class separability of the data in the embedded space, similar to DEC. The third term is introduced under the manifold regularization as explained in Section 2.4.1. The embedding that preserves the manifold structure of the data should be favored.

The formulation (3.1) is not jointly convex with respect to all the variables. The objective function is solved alternately between two groups of variables:

1) Fixing $\boldsymbol{\beta}$ and $\boldsymbol{G}$, and optimizing $\boldsymbol{F}$. Since $\boldsymbol{F}$ is constrained to be an indicator, the optimal $\boldsymbol{F}$ is

$$\boldsymbol{f}_{ij} = \begin{cases} 1, & j = \operatorname{argmin}_k \ \left\| \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} - \boldsymbol{g}_k^T \right\|_2^2 \\ 0, & \text{otherwise.} \end{cases} \quad \text{, for } i = 1, \ldots, n. \tag{3.2}$$

Using this updating formula, the solution goes quickly to the local optimum. There-

fore, the same update rules in DEC are used to avoid the local optimum problem: 1) run k-means on the embedded data $\bar{\boldsymbol{H}}\boldsymbol{\beta}$ with the initial cluster centroid equal to $\boldsymbol{G}$. The resulting cluster indicator $\boldsymbol{F}^{(b)}$ and the value of k-means objective function $o^{(b)}$ are saved as the baselines; 2) run k-means on the embedding $\bar{\boldsymbol{H}}\boldsymbol{\beta}$ for 20 times with randomly generated initialization. If the i-th obtained objective function reaches a lower value than the baseline, i.e., $o^{(i)} < o^{(b)}$, the cluster indicator matrix is updated using the corresponding results $\boldsymbol{F}^{(i)}$; otherwise, the baseline solution $\boldsymbol{F}^{(b)}$ is retained.

2) Fixing $\boldsymbol{F}$, and optimizing $\boldsymbol{\beta}$ and $\boldsymbol{G}$. The objective function becomes

$$L(\boldsymbol{\beta}, \boldsymbol{G}) = \quad \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) - \lambda \left\| \bar{\boldsymbol{H}}\boldsymbol{\beta} - \boldsymbol{F}\boldsymbol{G}^T \right\|_F^2 - \gamma \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{L} \bar{\boldsymbol{H}} \boldsymbol{\beta}\right). \quad (3.3)$$

Taking the derivative with respect to $\boldsymbol{G}$ first and setting it to zero,

$$\frac{\partial L}{\partial \boldsymbol{G}} = -2\lambda \left(\boldsymbol{G}\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{F}\right) = 0, \quad (3.4)$$

we have

$$\boldsymbol{G} = \boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{F} \left(\boldsymbol{F}^T \boldsymbol{F}\right)^{-1}. \quad (3.5)$$

Substituting $\boldsymbol{G}$ in Equation (3.5) into Equation (3.3), $L(\boldsymbol{\beta}, \boldsymbol{G})$ reduces to $L(\boldsymbol{\beta})$ as follows:

$$\begin{aligned}
L(\boldsymbol{\beta}) &= \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) - \lambda \left\| \bar{\boldsymbol{H}}\boldsymbol{\beta} - \boldsymbol{F}\left(\boldsymbol{F}^T\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T \bar{\boldsymbol{H}}\boldsymbol{\beta} \right\|_F^2 - \gamma \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{L} \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) \\
&= \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) - \lambda \operatorname{Tr}\left(\left(\bar{\boldsymbol{H}}\boldsymbol{\beta} - \boldsymbol{F}\left(\boldsymbol{F}^T\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T \bar{\boldsymbol{H}}\boldsymbol{\beta}\right)^T \left(\bar{\boldsymbol{H}}\boldsymbol{\beta} - \boldsymbol{F}\left(\boldsymbol{F}^T\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T \bar{\boldsymbol{H}}\boldsymbol{\beta}\right)\right) \\
&\quad - \gamma \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{L} \bar{\boldsymbol{H}} \boldsymbol{\beta}\right) \\
&= \operatorname{Tr}\left(\boldsymbol{\beta}^T \left(\bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} - \lambda \bar{\boldsymbol{H}}^T \bar{\boldsymbol{H}} + \lambda \bar{\boldsymbol{H}}^T \boldsymbol{F}\left(\boldsymbol{F}^T\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T \bar{\boldsymbol{H}} - \gamma \bar{\boldsymbol{H}}^T \boldsymbol{L} \bar{\boldsymbol{H}}\right)\boldsymbol{\beta}\right) \\
&= \operatorname{Tr}\left(\boldsymbol{\beta}^T \bar{\boldsymbol{H}}^T \boldsymbol{M} \bar{\boldsymbol{H}} \boldsymbol{\beta}\right),
\end{aligned} \quad (3.6)$$

where $\boldsymbol{M} = (1-\lambda)\boldsymbol{I} + \lambda\boldsymbol{F}\left(\boldsymbol{F}^T\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T - \gamma\boldsymbol{L}$. The optimization problem with respect to $\boldsymbol{\beta}$ becomes

$$\begin{aligned} \max_{\boldsymbol{\beta}} \quad & L(\boldsymbol{\beta}) \\ \text{s.t.} \quad & \boldsymbol{\beta}^T\boldsymbol{\beta} = \boldsymbol{I}_{n_O}. \end{aligned} \tag{3.7}$$

Notice that the maximum value of the objective function $L(\boldsymbol{\beta})$ is infinite if β can take an arbitrarily large value. Therefore, there is a need to put a constraint on the norm of $\boldsymbol{\beta}$. The global optimal solution to (3.7) can be derived by selecting the n_O normalized eigenvectors corresponding to the n_O largest eigenvalues of the eigenvalue problem:

$$\bar{\boldsymbol{H}}^T\boldsymbol{M}\bar{\boldsymbol{H}}\boldsymbol{v} = \mu\boldsymbol{v}. \tag{3.8}$$

Let $\mu_1, \mu_2, \ldots, \mu_{n_O}$ be the n_O largest eigenvalues of (3.8) and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n_O}$ be the corresponding normalized eigenvectors. Then, the solution to the output weights is obtained by

$$\boldsymbol{\beta}^* = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n_O}]. \tag{3.9}$$

Regarding the initialization of the iterative method, k-means is conducted on the data in the original space to get the initial values for cluster indicator $\boldsymbol{F}$. Then the output weights $\boldsymbol{\beta}$ are initialized using Equation (3.8) and (3.9) and initialize the cluster centroid matrix $\boldsymbol{G}$ using Equation (3.5). The proposed ELM-JEC is summarized in Algorithm 1.

3.3 Experimental Results

In this section, the performance of the proposed ELM-JEC algorithm is empirically compared with the related embedding and clustering methods on a wide range of real-world datasets including both image and non-image data.

Algorithm 1 The ELM-JEC algorithm

Require: The training data: $\boldsymbol{X} \in \mathbb{R}^{n \times n_I}$; the number of clusters: c; the trade-off parameters: λ, γ; the similarity graph: $\boldsymbol{W} \in \mathbb{R}^{n \times n}$; the dimensionality of embedded space: n_O.

Ensure: The embedding in a n_O-dimensional space: $\boldsymbol{Y} \in \mathbb{R}^{n \times n_O}$;
The cluster indicator matrix: $\boldsymbol{F} \in \mathbb{R}^{n \times c}$.

Initialization:

Step 1: Initialize an ELM network with n_H hidden neurons and calculate the centered output matrix of hidden layers $\bar{\boldsymbol{H}}$.

Step 2: Run k-means on the training data $\boldsymbol{X}$ to get the initial cluster indicator matrix $\boldsymbol{F}$.

Step 3: Initialize the output weights $\boldsymbol{\beta}$ using Equation (3.8) and (3.9).

Step 4: Initialize the cluster centroid matrix $\boldsymbol{G}$ using Equation (3.5).

Update:

Step 5: Run k-means with current cluster centroid matrix $\boldsymbol{G}$ and record the baseline cluster indicator matrix $\boldsymbol{F}^{(b)}$ and the baseline value $o^{(b)}$ of the k-means objective function. Run k-means with 20 randomly generated initialization. If the i-th obtained objective function value is smaller than the baseline value, i.e., $o^{(i)} < o^{(b)}$, update the cluster indicator using $\boldsymbol{F}^{(i)}$; otherwise, update using $\boldsymbol{F}^{(b)}$.

Step 6: Update output weights $\boldsymbol{\beta}$ using Equation (3.8) and (3.9)

Step 7: Update the cluster centroid matrix $\boldsymbol{G}$ using Equation (3.5).

Step 8: Go to *Step 5* until a maximum allowable number of iterations is reached or the label matrix $\boldsymbol{F}$ is the same as that in the previous iteration, which is the early stopping criterion.

Output:

Step 9: Calculate the embedding matrix $\boldsymbol{Y} = \bar{\boldsymbol{H}}\boldsymbol{\beta}$.

Return: $\boldsymbol{Y}$ as the embedding of the data; $\boldsymbol{F}$ as the cluster indication matrix.

Table 3.2: Specification of the benchmark datasets used to evaluate ELM-JEC

Data sets	# clusters	# features	# observations
Iris	3	4	150
Diabetes	2	8	768
Segment	7	19	2310
Isolet	26	617	7797
Pollen	7	625	630
UMIST	20	644	575
COIL20	20	1024	1440
ORL	40	1024	400

3.3.1 Datasets and Experimental Setting

The datasets used are summarized in Table 3.2. The first four are low-dimensional datasets taken from UCI Repository [60]. Isolet[1] is voice data set collected from 26 individuals. Pollen[2], UMIST[3], COIL20[4], and ORL[5] are image datasets [15, 11], where the raw pixel values are directly used as input features. Pollen contains pollen grain images, UMIST and ORL are face images datasets, and COIL20 contains 20 types of object images. The min-max normalization was used to normalize each feature to the range of [-1, 1].

The proposed ELM-JEC algorithm was compared with several related algorithms: k-means [61], LDA-KM [29], ELM clustering based on LDA (ELMC^{LDA}) [35], US-ELM [15], DEC [11]. k-means is one of the oldest clustering algorithms but still widely used. LDA-KM and DEC are non-ELM-related algorithms that conduct embedding and clustering jointly. Both as ELM-based algorithms, US-ELM has the property of structure-preserving and ELMC^{LDA} has the property of separability-maximizing. Clustering accuracy as defined in Section 4.3.1 was used as the evaluation metric.

[1] http://archive.ics.uci.edu/ml/datasets/isolet
[2] http://ome.grc.nia.nih.gov/iicbu2008/pollen/index.html
[3] http://images.ee.umist.ac.uk/danny/database.html
[4] http://www1.cs.columbia.edu/cave/research/softlib/coil-20.html
[5] http://www.uk.research.att.com/facedatabase.html

For a fair comparison, the experiment standardized the parameter settings, the initialization conditions, and the termination conditions of all methods as follows. The number of hidden neurons in the ELM-based methods was set to 1000 for Iris and 2000 for the other datasets. The sigmoid nonlinear activation function was used. The trade-off parameters in LDA-KM, ELMC^{LDA}, US-ELM, DEC, and ELM-JEC were selected from the same set, i.e., [2^{-4},2^{-3}, ..., 2^{4}]. The *similarity graph* used by US-ELM and ELM-JEC was constructed using the 5-nearest-neighbor graph with binary weights. The dimensionalities of embedded space used in US-ELM, DEC, and ELM-JEC were selected from a set of [2, 4, 8, 16, 32]. Iterative methods, i.e., LDA-KM and ELM-JEC, were initialized with outputs of k-mean and were terminated when there was no change in the predicted labels, or a maximum number of iteration (20 in our experiment) was reached. The updating rules for cluster indicator matrix in ELM-JEC and DEC were the same. All algorithms were run for ten times independently.

3.3.2 Clustering Accuracy in Comparison with Related Methods

Table 3.3: ACC(%) of ELM-JEC and the related algorithms on the benchmark datasets

Data Set	k-means	LDA-KM	ELMC^{LDA}	US-ELM	DEC	**ELM-JEC**
Iris	76.13 ± 16.18	88.80 ± 16.58	81.60 ± 12.70	91.07 ± 8.38	95.33 ± 0	**97.20 ± 0.28**
Diabetes	66.88 ± 1.25	67.46 ± 0.78	67.71 ± 0.09	65.89 ± 0.06	67.71 ± 0	**74.26 ± 1.11**
Segment	60.67 ± 7.83	**69.66 ± 7.90**	69.64 ± 4.53	66.10 ± 6.22	66.90 ± 0.07	68.95 ± 1.41
Isolet	53.21 ± 3.71	57.22 ± 4.45	54.36 ± 4.23	54.15 ± 4.09	56.30 ± 2.16	**57.63 ± 1.24**
Pollen	46.95 ± 2.16	45.98 ± 4.83	46.97 ± 3.72	48.35 ± 2.81	49.70 ± 0.05	**50.25 ± 1.35**
UMIST	42.31 ± 1.56	42.17 ± 3.52	42.78 ± 3.57	65.11 ± 6.09	44.38 ± 2.02	**68.02 ± 3.01**
COIL20	61.67 ± 4.33	60.15 ± 3.91	59.67 ± 4.21	78.69 ± 4.66	67.48 ± 2.54	**80.79 ± 2.56**
ORL	56.18 ± 3.21	50.38 ± 1.89	48.25 ± 3.11	**59.63 ± 1.96**	57.45 ± 2.66	58.38 ± 2.08

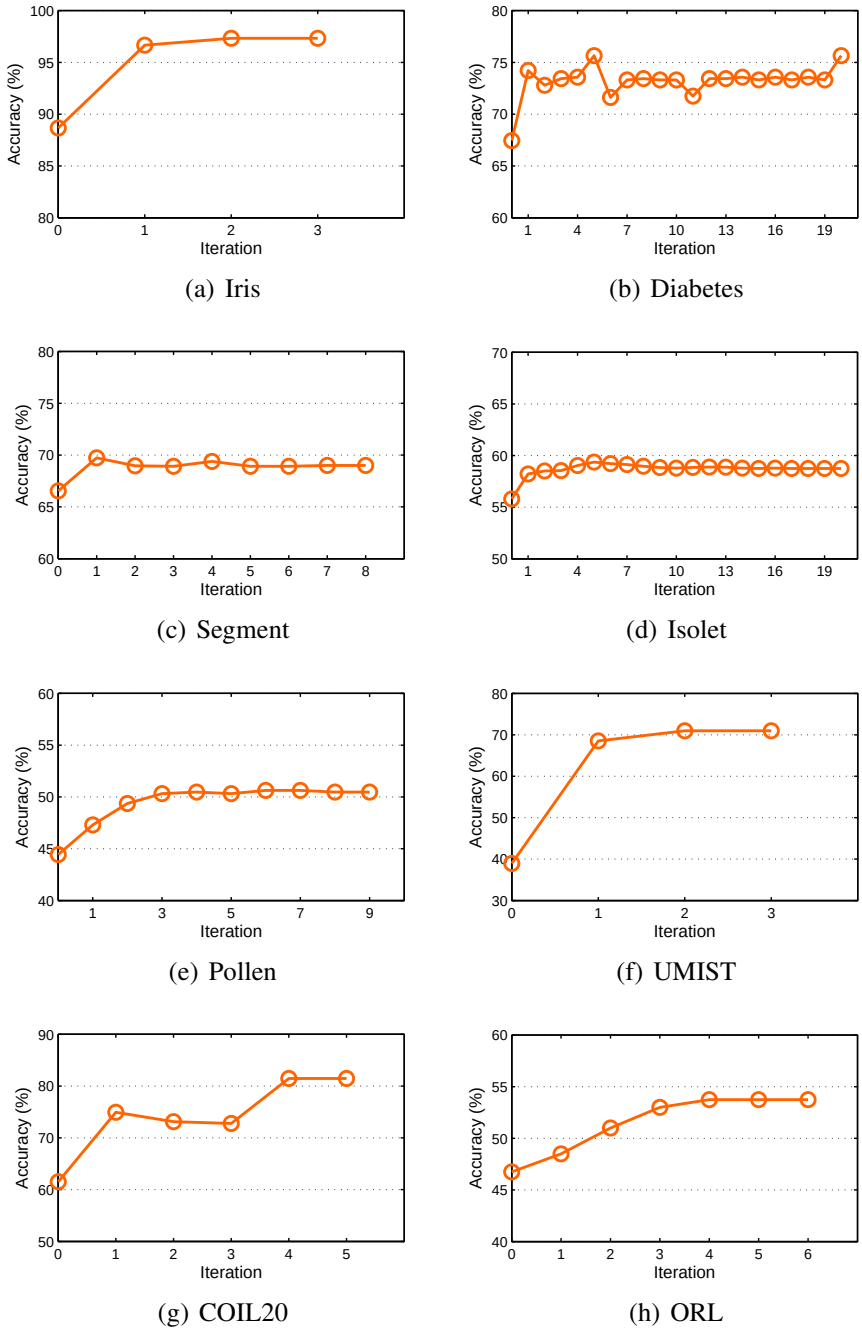
Table 3.3 presents the averaged clustering accuracy for all the algorithms. LDA-KM and ELMC^{LDA} outperformed the baseline method k-means on most datasets; but on image datasets, conducting k-means in the original feature space achieved higher or comparable accuracy. US-ELM achieved better results than DEC on human-face image datasets and object image datasets, showing the nonlinear embedding that preserves manifold structure is desired in complex image clustering tasks. DEC produced better results, compared with US-ELM on most non-image datasets and simple image dataset, i.e., Pollen (compared with the complex shapes of human faces and objects of different categories, pollen grains have relatively simple shapes). Compared with the other methods, ELM-JEC achieved favorable clustering accuracy on most of the datasets. Moreover, ELM-JEC showed smaller standard deviation than other ELM-based methods.

3.3.3 Clustering Accuracy over Iterations

This experiment studied how clustering accuracy changes over the iterations in ELM-JEC. As can be seen from Fig. 3.1, ELM-JEC was able to improve the clustering accuracy over iterations in general. For most of the datasets, ELM-JEC stoped within a small number of iterations (less than 10). On Diabetes and Isolet, the accuracy fluctuated within a small range after significant improvement was made at the ffirst iteration. In fact, a significant improvement was made within the first one or two iterations in most cases.

3.3.4 Influence of Dimensionality of Embedded Space

This experiment studied the influence of the dimensionality of the embedded space in three embedding methods, i.e., US-ELM, DEC, ELM-JEC. The clustering accuracies under different dimensionalities are reported in Fig. 3.2. As observed, ELM-JEC and DEC produced satisfactory results using a larger range of dimensionalities compared with US-ELM. Moreover, ELM-JEC produced comparable or even better results than



(a) Iris (b) Diabetes

(c) Segment (d) Isolet

(e) Pollen (f) UMIST

(g) COIL20 (h) ORL

Figure 3.1: Evolution of clustering accuracy of ELM-JEC as a function of iterations

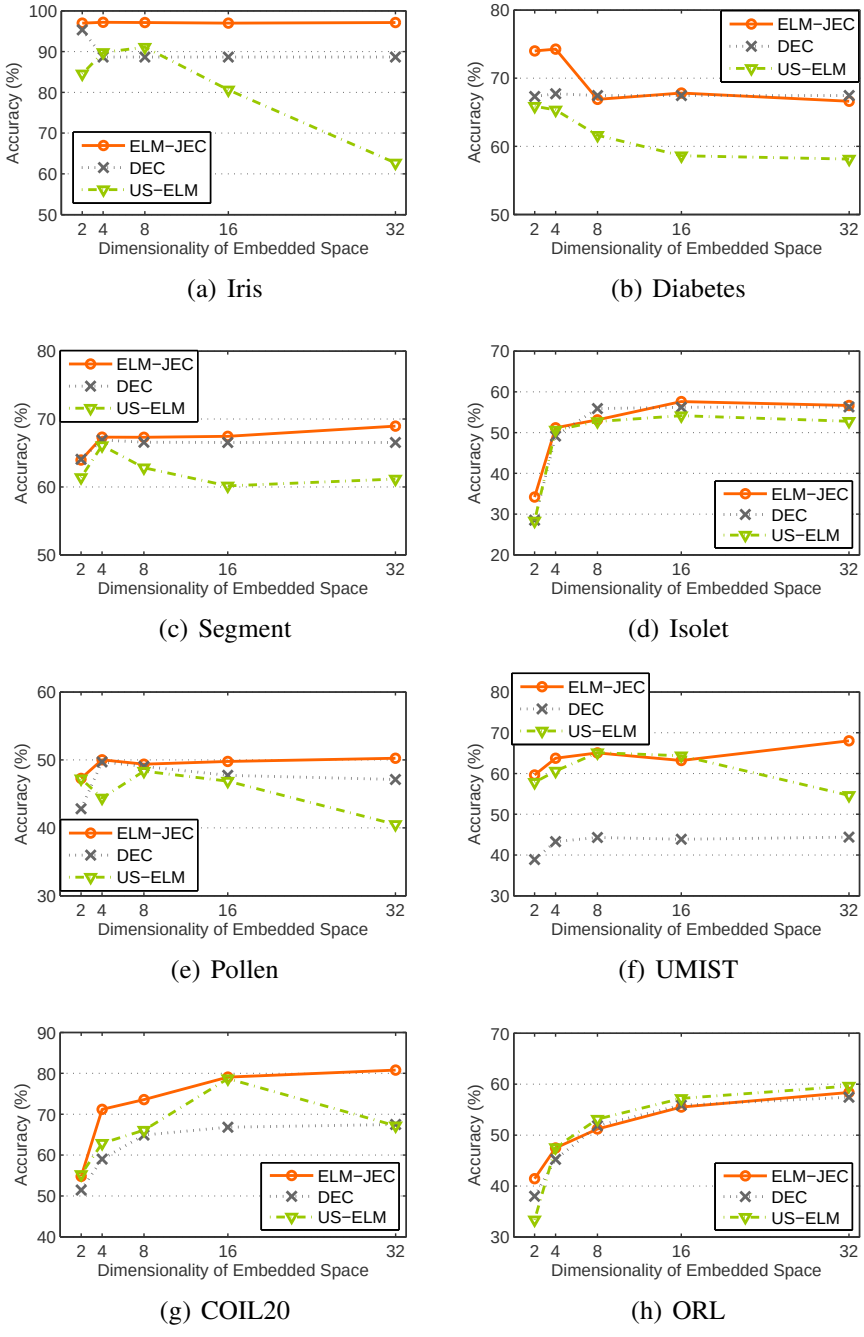
DEC on most datasets. This observation demonstrates that ELM-JEC is relatively stable and provides competitive performance even when the embedded dimensionality is not set to the optimal value, which makes it very suitable for applications where specific embedded dimensionality is required.

3.3.5 Influence of Trade-off Parameters

This experiment continued to investigate the influence of trade-off parameters used in ELM-JEC. Fig. 3.3 shows the clustering accuracy of ELM-JEC with different combinations of trade-off parameters values, which are under the respective optimal dimensionality of embedded space. On most of the datasets, high accuracy was observed over a large range of values. However, on Diabetes, Segment, and UMIST, favorable parameters were limited to a small range. On Ecoli, clustering accuracy fluctuated within a small range. Though ELM-JEC produced stable performance over a large range of embedded dimensionalities in most cases, reliable selection of trade-off parameters is need.

3.4 Summary

This chapter addresses the first research objective of the thesis, i.e., to develop effective clustering algorithms by taking advantage of ELM. Section 3.1 analyzes other embedding-based clustering methods, their advantages, and their limitations. To address these limitations, Section 3.2 proposes a joint embedding and clustering method based on ELM, called ELM-JEC. Section 3.3 demonstrates the advantage of ELM-JEC on a board range of benchmark datasets, i.e., it improves the clustering accuracy over iterations, it shows stable clustering performance over a relatively large range of embedded dimensionalities, and it achieves favorable performance compared with the related methods.



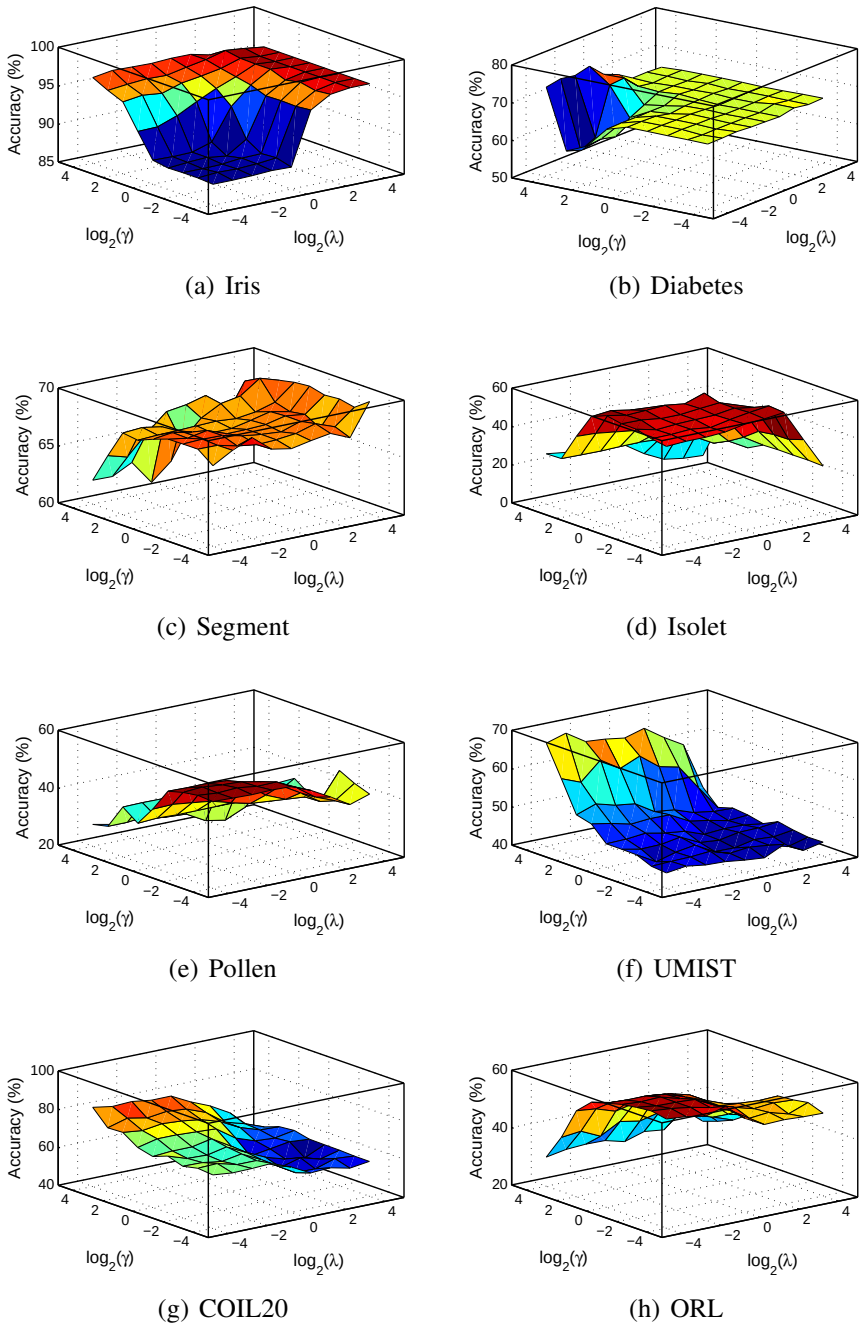
(a) Iris (b) Diabetes

(c) Segment (d) Isolet

(e) Pollen (f) UMIST

(g) COIL20 (h) ORL

Figure 3.2: Influence of the embedded space dimensionality on ELM-JEC, DEC, and US-ELM



(a) Iris (b) Diabetes

(c) Segment (d) Isolet

(e) Pollen (f) UMIST

(g) COIL20 (h) ORL

Figure 3.3: Influence of the trade-off parameters γ and λ on ELM-JEC

Chapter 4

Graph Learning based on Dual Data Representations for Clustering

Chapter 4 focuses on graph-based clustering methods. Existing adaptive graph learning methods learn the graph based on either original data or linearly projected data. To handle data with complex structures, this chapter develops dual representations and proposes to learn graph based on them. The proposed algorithm is then evaluated on general datasets.

4.1 Background and Motivation

Adaptive graph learning for clustering is an emerging clustering framework. As reviewed in Section 2.3.3, the objective of the three state-of-the-art algorithms, i.e., Clustering with Adaptive Neighbors (CAN), Projected Clustering with Adaptive Neighbors (PCAN), and Constrained Laplacian Rank (CLR), are to solve a graph that is consistent with the structure of the data and has c connected components (where c is the required number of clusters). Since the graph's having c connected components, the cluster indicators can be obtained directly from the graph without any post-processing, unlike traditional spectral clustering methods. To retain the consistency with the data structure, CAN forces the data similarity to be inversely proportional to the pairwise distance among the original data [16]. In PCAN, the pairwise distances in some linear subspace are used to adjust the similarity matrix, and the linear subspace is also updated based on the graph [16]. In CLR, the graph to be learned is constrained to have preferably little difference from an initial graph, which is constructed based on the original data [17].

Ideally, data points in the same cluster should be close to each other, and data points in different clusters should be far away. The graph learned based on distance can accurately capture the underlying similarity relationship. However, in real-world applications, due to noise and high dimensionality, data points may lie on a complex manifold, i.e., data points in the same cluster may also be far away – even farther away than those in different clusters [28]. In these cases, considering only local distances in the original space, CAN and CLR may suffer from the poor basis that is used to capture the data similarity. Though PCAN tries to project the data onto a linear subspace and adjusts the projection jointly with graph learning, it still has two problems: 1) it completely abandons data in the original space, which can also be good indicators to similarity relationship sometimes and 2) quite often, a linear projection cannot sufficiently discard the noise or capture the underlying nonlinear data structure. Based on the above observations, one way to improve the clustering performance is to learn the graph based on better representations of data, which could be achieved through nonlinear embeddings.

Neural networks are efficient tools to approximate nonlinear functions explicitly [62] and can be used to obtain the nonlinear embedding. To be applied as a preliminary step for graph learning, the neural network training algorithm should guarantee universal approximation capability and have high efficiency. As a training algorithm for "generalized" single hidden layer feedforward neural networks (SLFNs), Extreme Learning Machine (ELM) [20] has its core idea that the hidden neurons can be generated randomly and fixed without sacrificing the universal approximation capability; meanwhile, only the output weights connecting hidden layers and output nodes need to be tuned so high efficiency can be achieved. Therefore, it is very promising to use ELMs to obtain nonlinear embeddings for graph learning.

To further improve the clustering performance by taking advantage of ELM, this chapter proposes a novel adaptive graph learning method for clustering by adjusting the graph based on dual data representations: the original data and the nonlinear embedding obtained using an ELM-based algorithm. Specifically, it is proposed to obtain the nonlinear ELM-based embedding by requiring it to preserve the pair-wise similarity relationship and form uncorrelated features in the embedded space. Meanwhile, the recently proposed method of constraining the rank of the graph Laplacian matrix [17] is used to force the graph to have the desired number of connected components as required by the clustering task.

4.2 Proposed Method

In a clustering task, a set of unlabeled data as $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{n}, x_i \in \mathbb{R}^{n_I}$ is given and the aim is to group the data into c clusters. The similarity graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of vertices and edges, respectively. Each vertex represents a data point, and the edge between a pair of data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is weighted by their similarity w_{ij}. A subset $\mathcal{A}$ is referred to as a component of the graph if it is connected, and there is no connection between any vertices in $\mathcal{A}$ and vertices in $\bar{\mathcal{A}}$.

To learn the similarity matrix $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n] = [w_{ij}]_{i=1,j=1}^{n}$ properly, two important criteria are considered and listed as follows: 1) the graph should form clear clusters (i.e., connected components) of the exact required number and 2) the weight should indicate the similarity between a pair of data points. To achieve the first objective, the recently proposed method of CLR [16, 17] is adopted. To achieve the second objective, the dual data representations, i.e., the original data and the nonlinear embedding obtained using a variant of ELMs, are adopted.

4.2.1 Graph Learning

Before explaining graph learning, the ELM-based nonlinear embedding and distances are first introduced for better clarity:

Embedding. Given an ELM network, the nonlinear embedding of a data set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{n}$ by ELM is denoted as $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^{n}$ and has the following form:

$$\boldsymbol{y}_i = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}, \tag{4.1}$$

where $\boldsymbol{h}(\boldsymbol{x})$ is the output of ELM hidden layer calculated based on $\boldsymbol{x}$ as explained in Section 2.2 and $\boldsymbol{\beta} \in \mathbb{R}^{n_H \times n_O}$ represents the output weights of ELM obtained using a proposed algorithm, which is explained in the following Section 4.2.3.

Distances. The pair-wise distance d_{ij} in the original feature space and the pair-wise distance p_{ij} in the embedded space are defined as:

$$\begin{aligned} d_{ij} &= \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \\ p_{ij} &= \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2. \end{aligned} \tag{4.2}$$

In the task of clustering, the distance between two data points sometimes cannot reveal the dissimilarity between them. Fig. 4.1 shows an example. Both $\boldsymbol{x}_3$ and $\boldsymbol{x}_7$ have a distance of 81 from $\boldsymbol{x}_6$. However, if the desired number of clusters is two, it is sensible

to partition the seven data points into two sets of $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ and $\{\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7\}$. In this sense, $\boldsymbol{x}_3$ is more dissimilar with $\boldsymbol{x}_6$ than $\boldsymbol{x}_7$. Another way to interpret the dissimilarity is that all the other points are even farther away from $\boldsymbol{x}_7$ than $\boldsymbol{x}_6$, while only one point is farther from $\boldsymbol{x}_3$ than $\boldsymbol{x}_6$. Therefore, a new dissimilarity measure is needed to put the pairwise distances into the perspective of both data points.

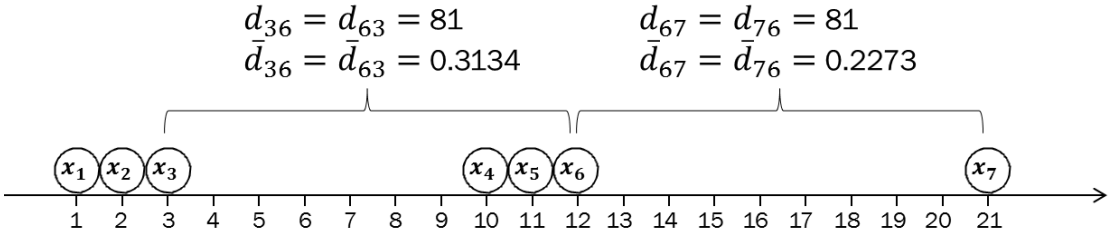


Figure 4.1: A case where the normalized distances better reveal the underlying cluster structure.

A new dissimilarity measure is proposed as the geometric mean of fractions of distance vector norms and is called the *normalized pairwise distance*. Mathematically, the normalized pairwise distances in the original feature space and the normalized pairwise distances in the embedded spaces are defined as:

$$\bar{d}_{ij} = \sqrt{\frac{d_{ij}}{\|\boldsymbol{d}_i\|_2} \cdot \frac{d_{ji}}{\|\boldsymbol{d}_j\|_2}}, \tag{4.3}$$

$$\bar{p}_{ij} = \sqrt{\frac{p_{ij}}{\|\boldsymbol{p}_i\|_2} \cdot \frac{p_{ji}}{\|\boldsymbol{p}_j\|_2}}, \tag{4.4}$$

where $\boldsymbol{d}_i = [d_{i1}, d_{i2}, \ldots, d_{in}]^\top$ is a vector containing distances between the i-th data point and the whole population in the original space, and $\boldsymbol{p}_i = [p_{i1}, p_{i2}, \ldots, p_{in}]^\top$ is a vector containing distances in the embedded space. The proposed normalized distances have the following properties:

Theorem 4.1 *The normalized distance is symmetric.*

Proof:

$$\bar{d}_{ij} = \sqrt{\frac{d_{ij}}{\|\boldsymbol{d}_i\|_2} \cdot \frac{d_{ji}}{\|\boldsymbol{d}_j\|_2}} = \sqrt{\frac{d_{ij}}{\|\boldsymbol{d}_j\|_2} \cdot \frac{d_{ji}}{\|\boldsymbol{d}_i\|_2}} = \bar{d}_{ji}. \tag{4.5}$$

The same proof applies in the embedded space $\bar{p}_{ij} = \bar{p}_{ji}$. ∎

Theorem 4.2 *If two data points are of equal distance to the third point, the ratio of their normalized distances is the square root of the inverse ratio of their respective distance vector norms.*

Proof: Suppose the i-th data point and the j-th data point are of equal distance with data point k, i.e., $d_{ik} = d_{jk}$. The ratio of the normalized distances is

$$\frac{\bar{d}_{ik}}{\bar{d}_{jk}} = \frac{\sqrt{\frac{d_{ik}}{\|\boldsymbol{d}_i\|_2} \cdot \frac{d_{ki}}{\|\boldsymbol{d}_k\|_2}}}{\sqrt{\frac{d_{jk}}{\|\boldsymbol{d}_j\|_2} \cdot \frac{d_{kj}}{\|\boldsymbol{d}_k\|_2}}} = \sqrt{\frac{\|\boldsymbol{d}_j\|_2}{\|\boldsymbol{d}_i\|_2}}. \tag{4.6}$$

The same proof applies in the embedded space $\frac{\bar{p}_{ik}}{\bar{p}_{jk}} = \sqrt{\frac{\|\boldsymbol{p}_j\|_2}{\|\boldsymbol{p}_i\|_2}}$. ∎

The advantages of the normalized pairwise distances are also demonstrated in Fig. 4.1: 1) The normalized pairwise distance takes into account the distances between the two data points and the whole population, and as a result, according to the normalized distance measure, $\boldsymbol{x}_3$ is much farther from $\boldsymbol{x}_6$ than $\boldsymbol{x}_7$, which is consistent with the underlying cluster structure; 2) The normalized pairwise distance has the symmetry property.

With the notation of embedding and distances, the mathematical formulation of the proposed ELM-CLR is described as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\bar{d}_{ij}\bar{p}_{ij}w_{ij} + \gamma w_{ij}^2\right) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1, \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{4.7}$$

where $\boldsymbol{L}$ is the graph Laplacian as defined in Section 2.4.1.

The first term in the objective function is to force the weight to be inversely proportional to the product of the normalized pair-wise distances in the original feature space and the embedded space. The second term in the objective function is added to avoid the trivial solution where only the weight corresponding to the nearest data point is 1 and the rest are all 0. $\gamma > 0$ is a trade-off parameter, and how to specify this parameter is explained in Section 4.2.2. The first and second constraints are introduced based on the basic properties of graph weights, i.e., the weights should be non-negative, and the sum of the weights of the edges from one vertex should be to one. The method of constrained Laplacian rank [16] is adopted, and an additional third constraint is added to the problem such that the graph learned has naturally c connected components corresponding to c clusters. A graph satisfying the third constraint naturally has c connected components, because the multiplicity c of the eigenvalue 0 of the Laplacian matrix $\boldsymbol{L}$ is equal to the number of connected components in the graph with the similarity matrix $\boldsymbol{W}$ [63].

Let $\sigma_i(\boldsymbol{L})$ denote the i-th smallest eigenvalue of $\boldsymbol{L}$, with $\sigma_i(\boldsymbol{L}) \geq 0$ as $\boldsymbol{L}$ is positive semidefinite. Problem (4.7) is approximated by the following problem with a sufficiently large value of λ:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\bar{d}_{ij}\bar{p}_{ij}w_{ij} + \gamma w_{ij}^2\right) + 2\lambda \sum_{i=1}^{c} \sigma_i(\boldsymbol{L}) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1 \end{aligned} \tag{4.8}$$

The parameter $\lambda > 0$ controls the trade-off between low rank of graph Laplacian and high consistency with the data structure; the desired rank of the graph Laplacian is generally reached with a large value of λ. According to the Ky Fan's Theorem [64], the

problem (4.8) can be transformed into the following form:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{E}} \quad & \sum_{i,j=1}^{n} \left(\bar{d}_{ij}\bar{p}_{ij}w_{ij} + \gamma w_{ij}^2\right) + 2\lambda \operatorname{Tr}\left(\boldsymbol{E}^T\boldsymbol{L}\boldsymbol{E}\right) \\ \text{s.t.} \quad & \forall i,j, w_{ij} \geq 0, \boldsymbol{w}_i^T\mathbf{1}_n = 1, \\ & \boldsymbol{E} \in \mathbb{R}^{n\times c}, \boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{I}_c \end{aligned} \tag{4.9}$$

This problem can be solved by means of alternating optimization method. During each iteration, the objective function is optimized with respect to $\boldsymbol{E}$ and $\boldsymbol{W}$ alternatively (while fixing the other) until the Laplacian rank constraint in the original problem is satisfied. The initialization of the graph will be described separately in Section 4.2.2. Here the alternating optimization process is first introduced:

In each iteration, $\boldsymbol{E}$ is first solved while $\boldsymbol{W}$ is fixed. The problem is re-written as:

$$\begin{aligned} \min_{\boldsymbol{E}} \quad & \operatorname{Tr}\left(\boldsymbol{E}^T\boldsymbol{L}\boldsymbol{E}\right) \\ \text{s.t.} \quad & \boldsymbol{E} \in \mathbb{R}^{n\times c}, \boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{I}_c \end{aligned} \tag{4.10}$$

The optimal solution of $\boldsymbol{E}$ can be formed by taking the c eigenvectors corresponding to the c smallest eigenvalues of $\boldsymbol{L}$.

Then $\boldsymbol{E}$ is fixed and $\boldsymbol{W}$ is updated. Also it can be verified that $\sum_{i,j=1}^{N} \|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2^2 w_{ij} = 2\operatorname{Tr}\left(\boldsymbol{E}^T\boldsymbol{L}\boldsymbol{E}\right)$, given $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]^T$, where $\boldsymbol{e}_i \in \mathbb{R}^c$. The problem becomes:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\bar{d}_{ij}\bar{p}_{ij}w_{ij} + \gamma w_{ij}^2\right) + \lambda\|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2^2 w_{ij} \\ \text{s.t.} \quad & \forall i,j, w_{ij} \geq 0, \boldsymbol{w}_i^T\mathbf{1}_n = 1 \end{aligned} \tag{4.11}$$

In the first term, the normalized pair-wise distance in the embedded space $\bar{p}_{ij}$ is calculated using the embedding achieved by an ELM network. The process of learning an ELM network is separately described in Section 4.2.3 for better clarity. It is worth noting that the learning of ELM network relies on the similarity matrix of the graph.

Since the graph is refined in each iteration, instead of fixing the ELM network to the one obtained using the initial similarity matrix, it is proposed in this section to learn a new ELM network and calculate the normalized pair-wise distances in each iteration before solving $\boldsymbol{W}$.

With the normalized pair-wise distance in the embedded space ready, the problem can be simplified by denoting $\boldsymbol{q}_i \in \mathbb{R}^n$ as a vector with the j-th element as $q_{ij} = \bar{d}_{ij}\bar{p}_{ij} + \lambda\|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2^2$. Moreover, the problem is independent between different $i \in [1, \ldots, n]$ and thus can be re-written into n individual problems with the i-th problem as

$$\begin{aligned} \min_{\boldsymbol{w}_i} \quad & \left\| \boldsymbol{w}_i + \frac{1}{2\gamma}\boldsymbol{q}_i \right\|_2^2 \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1 \end{aligned} \tag{4.12}$$

Here this problem can be solved with a closed form solution in the same way as in [16].

After obtaining the updated $\boldsymbol{E}$ and $\boldsymbol{W}$ in the current iteration, the algorithm proceeds in three manners according to the clustering capability of the graph:

(1). The Graph Has Fewer Connected Components Than Required. Since the initial graph is usually a fully connected graph and a small value of λ in most cases, the iterative process follows a gradual increase in the number of connected components. In this case of insufficient connected components, i.e., $\mathrm{rank}(\boldsymbol{L}) > n - c$, the same heuristic of increasing λ [16], i.e., multiplying λ by 2, is used to enforce the low rank of graph Laplacian term. Then the algorithm goes to the next iteration.

(2). The Graph Has More Connected Components Than Required. The overrun of the number of connected components, i.e., $\mathrm{rank}(\boldsymbol{L}) < n - c$, is undesirable and could be caused by a large value of λ. Following the same heuristic [16], the similarity matrix is overwritten using the values obtained in the previous iteration; at the same time, λ is divided by 2 in order to avoid the situation of overrun number of connected components. Then the algorithm goes to the next iteration.

(3). The Number of Connected Components of the Graph Is as Required. The algorithm stops in this case and returns the graph with exact c connected components.

As can be seen from the above three cases, the algorithm stops only if the desired number of connected components is achieved. This condition may not be satisfied even after many iterations, so an early stopping criterion is added that the algorithm stops if a user-defined maximum number of iteration is reached. Based on the observations on the test datasets, the algorithm converges in most cases. After investigation, we found the main reason for non-convergence is due to the initial graph's having more connected components than desired, which is the result of inappropriate hyper-parameter values. In the case of non-convergence, we suggest re-running the algorithm after changing the hyper-parameter values according to the following guidelines: 1) increase the number of neighbors k, which is introduced in Section 4.2.2, 2) increase the weight on the non-graph-related term in ELM network learning, i.e., the regularization parameter δ, which is introduced in Section 4.2.3, 3) reduce the dimensionality of the embedded space n_O. These guidelines may also apply to other adaptive graph learning methods, like CAN and PCAN.

4.2.2 Graph Initialization

This section describes how to initialize the similarity matrix $\boldsymbol{W}$ of the graph based on only the data in the original feature space. Intuitively, a small normalized pair-wise distance $\bar{d}_{ij}$ in the original features space should be associated with a large similarity weight w_{ij}. The initial graph learning problem is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\bar{d}_{ij} w_{ij} + \gamma w_{ij}^2\right) \\ \text{s.t.} \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^T \mathbf{1}_n = 1 \end{aligned} \tag{4.13}$$

The problem can be written in the same form as problem (4.12) with $q_{ij} = \bar{d}_{ij}$, which can be solved with a closed form solution in the same way as in [16].

Here instead of specifying the parameter $\gamma \in \mathbb{R} > 0$, the value of γ can be calculated based on the desired number of neighbors $k \in \mathbb{N} > 0$ in the same way as CAN [16]:

$$\gamma = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{k}{2}\bar{d}^s_{i,k+1} - \frac{1}{2}\sum_{j=1}^{k}\bar{d}^s_{ij}\right), \tag{4.14}$$

where $\bar{d}^s_{i1}, \bar{d}^s_{i2}, \ldots, \bar{d}^s_{in}$ are sorted normalized distances in the original space from small to large. As k is an integer with explicit meaning, it is easier to tune. Meanwhile, we use the value obtained here in graph initialization stage to set the value of γ in (4.11). The other hyper parameter λ in (4.11) is updated during the graph learning, and we can specify its initial value with a guess, e.g., using the value of γ.

4.2.3 ELM Network Learning

Given the set of data $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ and the similarity matrix $\boldsymbol{W} = [w_{ij}]$, this section aims to obtain an embedding that preserves the pair-wise data similarity and forms uncorrelated features in the embedded space. For a large similarity weight w_{ij}, the i-th and j-th data points should be close in the embedded space as measured by a small Euclidean distance, which can be formulated mathematically as

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \sum_{i,j=1}^{n} \|\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} - \boldsymbol{h}(\boldsymbol{x}_j)\boldsymbol{\beta}\|_2^2 w_{ij} + \delta\|\boldsymbol{\beta}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^T \boldsymbol{S}_t^H \boldsymbol{\beta} = \boldsymbol{I}_{n_O} \end{aligned} \tag{4.15}$$

where $\boldsymbol{S}_t^H = \bar{\boldsymbol{H}}^T\bar{\boldsymbol{H}} \in \mathbb{R}^{n_H \times n_H}$ is the total scatter matrix of the data in the ELM hidden neuron space, and $\bar{\boldsymbol{H}} = \left(\boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\right)\boldsymbol{H}$ is the mean centred matrix of the hidden neuron output matrix $\boldsymbol{H}$. Here the second term in Equation (4.15) is a regularization term against over-fitting and $\delta > 0$ is a user-defined parameter.

This section proposes to constrain the covariance matrix of the embedding to be the identity matrix in order to avoid the trivial solution of $\boldsymbol{\beta}$ where all elements are zero.

Compared with US-ELM [15], which has the same objective function but a different constraint of the form $(\boldsymbol{H}\boldsymbol{\beta})^T\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{I}_{n_O}$, the embedding obtained in the proposed formulation have two desirable properties [16, 65] that 1) the features are less correlated with each other and 2) all features tend to have unit variance. To solve the optimization problem, two cases are considered:

1) The Number of Data Points Is Larger Than the Number of Hidden Neurons

The optimization problem (4.15) can be represented efficiently in the matrix form as follows:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \mathrm{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{\beta}\right) + \delta\|\boldsymbol{\beta}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^T\boldsymbol{S}_t^H\boldsymbol{\beta} = \boldsymbol{I}_{n_O} \end{aligned} \tag{4.16}$$

The optimal solution to problem (4.16) is given by choosing $\boldsymbol{\beta}$ as the matrix whose columns are the eigenvectors (normalized to satisfy the constraint) corresponding to the first n_O smallest eigenvalues of the following generalized eigenvalue problem:

$$\left(\delta\boldsymbol{I}_{n_H} + \boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\right)\boldsymbol{\mu} = \tau\bar{\boldsymbol{H}}^T\bar{\boldsymbol{H}}\boldsymbol{\mu}. \tag{4.17}$$

Let $\tau_1, \tau_2, \ldots, \tau_{n_O}$ be the n_O smallest eigenvalues in ascending order of Equation (4.17) and $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_{n_O}$ be their corresponding eigenvectors. Then, the solution to the output weight $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}^* = [\bar{\boldsymbol{\mu}}_1, \bar{\boldsymbol{\mu}}_2, \ldots, \bar{\boldsymbol{\mu}}_{n_O}], \tag{4.18}$$

where $\bar{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i / \left\|\bar{\boldsymbol{H}}\boldsymbol{\mu}_i\right\|_2, i = 1, \ldots, n_O$ are the normalized eigenvectors.

2) The Number of Data Points Is Not Larger than the Number of Hidden Neurons

In this case, Problem (4.17) is underdetermined. The common approach to handle this problem is to restrict $\boldsymbol{\beta}$ to be a linear combination of the rows of $\boldsymbol{H}$: $\boldsymbol{\beta} = \boldsymbol{H}^T\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in \mathbb{R}^{n\times n_O}$). According to this approach, instead of solving $\boldsymbol{\beta}$ of size $n_H \times n_O$, the task becomes solving a smaller matrix $\boldsymbol{\alpha}$ of size $n \times n_O$ ($n < n_H$). Substituting $\boldsymbol{\beta} = \boldsymbol{H}^T\boldsymbol{\alpha}$ into (4.16), we get

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \mathrm{Tr}\left(\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{\alpha}\right) + \delta\|\boldsymbol{H}^T\boldsymbol{\alpha}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{S}_t^H\boldsymbol{H}^T\boldsymbol{\alpha} = \boldsymbol{I}_{n_O} \end{aligned} \tag{4.19}$$

The optimal solution to Problem (4.19) is given by choosing $\boldsymbol{\alpha}$ as the matrix whose columns are normalized eigenvectors corresponding to the first n_O smallest eigenvalues of the generalized eigenvalue problem:

$$\left(\delta\boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{H}\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)\boldsymbol{\nu} = \rho\boldsymbol{H}\bar{\boldsymbol{H}}^T\bar{\boldsymbol{H}}\boldsymbol{H}^T\boldsymbol{\nu}, \tag{4.20}$$

where $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_{n_O}$ are the generalized eigenvectors corresponding to the n_O smallest eigenvalues $\rho_1, \rho_2, \ldots, \rho_{n_O}$. To further simplify the problem, multiplying both side of Equation (4.20) from the left by $\left(\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}$, we get

$$\left(\delta\boldsymbol{I}_n + \boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)\boldsymbol{\nu} = \rho\boldsymbol{M}\boldsymbol{\nu}, \tag{4.21}$$

where $\boldsymbol{M} = \left(\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{H}\bar{\boldsymbol{H}}^T\bar{\boldsymbol{H}}\boldsymbol{H}^T$.

The eigenvectors can be obtained by solving Problem (4.21) as a generalized eigenvalue problem. Then the resulting eigenvectors are further normalized to satisfy the constraint in Problem (4.20). The final solution to output weight $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta}^* = \boldsymbol{H}^T[\bar{\boldsymbol{\nu}}_1, \bar{\boldsymbol{\nu}}_2, \ldots, \bar{\boldsymbol{\nu}}_{n_O}], \tag{4.22}$$

Algorithm 2 The ELM-CLR Algorithm

Require: The training data $\mathcal{X}$, the number of clusters c, the number of neighbors k, the regularization parameter δ, the dimensionality of embedded space n_O, the number of hidden neurons n_H;

Ensure: The similarity matrix $\boldsymbol{W}$ with exact c connected components.

Step 1: Compute the normalized pairwise distance in the original space $\{\bar{d}_{ij}\}_{i=1,j=1}^{n}$ according to Equation (4.3) and initialize the similarity matrix $\boldsymbol{W}$ using the optimal solutions to (4.13).

Step 2: Compute γ and the initial value of λ according to Equation (4.14).

Step 3: Initialize an ELM network of n_H hidden neurons with random input weights and biases, and calculate the output of hidden neurons.

Step 4: Update the output weight $\boldsymbol{\beta}$ in ELM network according to Equation (4.18) or (4.22), and compute the normalized pairwise distances in the ELM embedded space $\{\bar{p}_{ij}\}_{i=1,j=1}^{n}$ according to Equation (4.4).

Step 5: Update the variable $\boldsymbol{F}$ using the optimal solution to (4.10), i.e., c normalized eigenvectors of $\boldsymbol{L}$ corresponding to the c smallest eigenvalues.

Step 6: Record the current similarity matrix as $\boldsymbol{W}_0$ and update the similarity matrix $\boldsymbol{W}$ using the optimal solution to (4.12).

Step 7: Check the rank of graph Laplacian $\boldsymbol{L}$

- **If** $\mathrm{rank}(\boldsymbol{L}) > n - c$
- Multiply λ by 2,
- Go to **Step 4**;
- **Else If** $\mathrm{rank}(\boldsymbol{L}) < n - c$
- Overwrite the similarity matrix $\boldsymbol{W}$ with $\boldsymbol{W}_0$
- Divide λ by 2,
- Go to **Step 4**;
- **Else**
- **Return** The similarity matrix $\boldsymbol{W}$ with exact c connected components.

where $\bar{\boldsymbol{\nu}}_i = \boldsymbol{\nu}_i / \left\| \bar{\boldsymbol{H}} \boldsymbol{H}^T \boldsymbol{\nu}_i \right\|_2, i = 1, \ldots, n_O$ are the normalized eigenvectors.

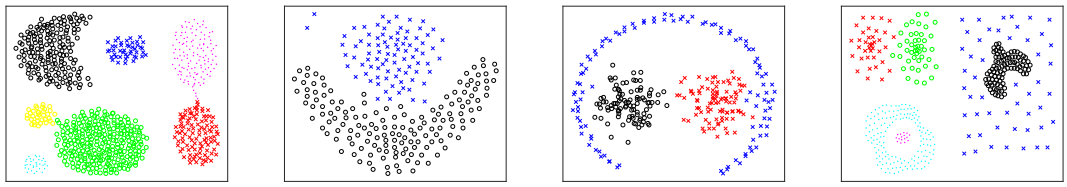
The complete ELM-CLR algorithm is described in Algorithm 2.

4.3 Experimental Results

This section evaluates the proposed algorithm on both synthetic and real-world clustering benchmark datasets in comparison with several related state-of-the-art algorithms.

4.3.1 Data Sets and Experimental Setup

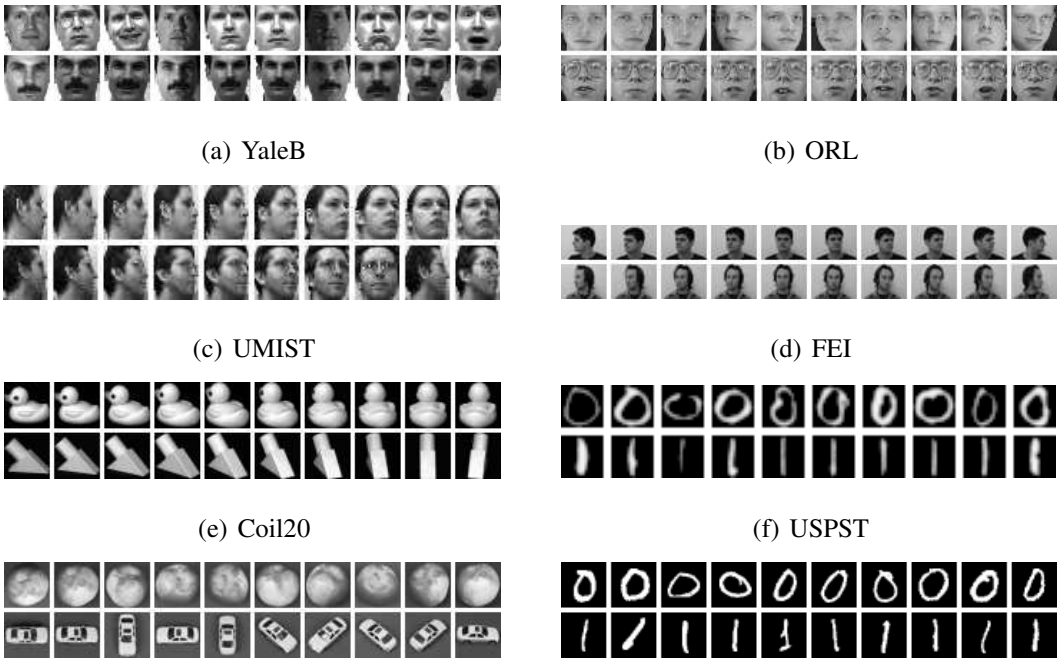
The proposed algorithm was evaluated on a wide range of benchmark datasets ranging from synthetic datasets to real-world datasets. The four well-known synthetic datasets[1], including Aggregation [66], Flame [67], Pathbased [68], and Compound [69], are two-dimensional datasets consisting of clusters of diverse shapes. Figure 4.2 presents the scatter plots of the datasets. The Aggregation dataset consists of seven clusters with no obvious circular clusters. The Flame, Pathbased, and Compound datasets contain circular clusters with decreasing amount of opening. (For example, the Pathbased dataset consists of a circular clustering with a small opening near the bottom, while the circular clusters in the Compound dataset enclose other clusters leaving no opening.) Such enclosure structure increases the difficulty by forming a highly nonlinear separation between clusters. Other challenges in these datasets include the density variation (e.g., in the Compound dataset), the shape variation (e.g., in the Aggregation dataset), and the noise which reduces the separation between clusters (e.g., in the Flame and Pathbased).



(a) Aggregation (b) Flame (c) Pathbased (d) Compound

Figure 4.2: Visualization of the synthetic data set. (The color coding represents the true labels.)

[1] https://cs.joensuu.fi/sipu/datasets/



(a) YaleB (b) ORL

(c) UMIST (d) FEI

(e) Coil20 (f) USPST

(g) ETH-80 (h) MNIST05

Figure 4.3: Sample images from the image datasets. (Each row is consist of 10 samples from one class.)

The real-world datasets included four UCI [60] datasets (Iris, Wine, Glass, and Ecoli), four face recognition image datasets (YaleB[2], ORL[3], UMIST[4], FEI[5]), and four object/digit recognition image datasets (COIL20, USPST, ETH-80[6], MNIST05). Figure 4.3 shows samples of widely-used image benchmark datasets [15, 70]. The Columbia University Image Library[7] dataset of 20 classes (COIL20) is an object classification dataset; USPST dataset is a subset (the testing set) of the handwriting recognition dataset USPS[8]; MNIST05 is a subset of the MNIST[9] handwritten digit database with sampling rate of 5%. The details of all datasets are summarized in Table 4.1.

[2]http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

[3]http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

[4]https://www.sheffield.ac.uk/eee/research/iel/research/face

[5]http://fei.edu.br/~cet/facedatabase.html

[6]https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/object-recognition-and-scene-understanding/analyzing-appearance-and-contour-based-methods-for-object-categorization/

[7]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[8]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps

[9]http://www.escience.cn/people/fpnie/papers.html

Table 4.1: Specifications of the benchmark datasets used to evaluate ELM-CLR

Type	Datasets	# clusters	# features	# observations
Synthetic	Flame	2	2	240
	Pathbased	3	2	300
	Compound	6	2	399
	Aggregation	7	2	788
UCI	Iris	3	4	150
	Wine	3	13	178
	Glass	6	9	214
	Ecoli	8	7	336
Image	YaleB	15	1024	165
	ORL	40	1024	400
	UMIST	20	644	575
	FEI	50	768	700
	COIL20	20	1024	1440
	USPST	10	256	2007
	ETH-80	8	1024	3280
	MNIST05	10	784	3495

To measure the quality of clustering, this study adopted the external validation approach, i.e., assessing the goodness of clustering solutions against the ground truth class labels. The evaluation metric was the widely-adopted clustering ACCuracy (ACC). Clustering ACC measures the percentage of correctly classified data points in the clustering solution compared with the ground-truth class labels and is defined as:

$$ACC = \frac{\sum_1^n \delta\left(t_i, \text{map}(t_i^p)\right)}{n}, \tag{4.23}$$

where n is the size of the dataset, t_i is the ground truth class label for the i-th data point, t_i^p is the cluster label in the clustering solution, map$(\cdot)$ is the optimal permuting function that maps each cluster label in the clustering solution to a ground truth class label, and $\delta(a, b) = 1$ if $a = b$, otherwise, $\delta(a, b) = 0$.

The proposed algorithm ELM-CLR was compared with the k-means, Spectral Clustering (SC) [71], Normalized Cut (NCut) [12], US-ELM [15], CAN [16], CLR [17], and PCAN [16]. k-means is one of the oldest clustering algorithms and still widely used [2]. SC and NCut are two famous and widely-used graph-based clustering algorithms. US-ELM is the first ELM-based algorithm that can be used for clustering problem. CAN, CLR, and PCAN are recently proposed adaptive graph learning methods for clustering problem. In addition, two variants of ELM-CLR were also implemented. The first one is denoted as ELM-CLR$^{\text{unnor}}$, where the graph learning is based on unnormalized distances and the other parts of the algorithm are the same as ELM-CLR. The objective function is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(d_{ij} p_{ij} w_{ij} + \gamma w_{ij}^2 \right) \\ s.t. \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^\top \mathbf{1}_n = 1 \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{4.24}$$

The second one is denoted as ELM-CLR$^{\text{add}}$, where the two types of normalized distances are combined using an addition operation, and the rest parts of the algorithm are the same as ELM-CLR. The objective function is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \sum_{i,j=1}^{n} \left(\left(\bar{d}_{ij} + \bar{p}_{ij} \right) w_{ij} + \gamma w_{ij}^2 \right) \\ s.t. \quad & \forall i, j, w_{ij} \geq 0, \boldsymbol{w}_i^\top \mathbf{1}_n = 1 \\ & \operatorname{rank}(\boldsymbol{L}) = n - c \end{aligned} \tag{4.25}$$

The parameter settings for all the algorithms were standardized as below:

• In the algorithms involving k-means, i.e., k-means, SC, NCut, and US-ELM, k-means were run for 100 times.

• The number of eigenvectors used in SC and NCut and the dimensionality of embedded space in PCAN, USELM, and ELM-CLR were selected from [2, 4, 8, 16, 32] $\cup$ $[c]$, where c is the number of clusters.

• The algorithms based on ELM, i.e., US-ELM and ELM-CLR 1) used sigmoid function as the activation function, 2) repeated the random initialization for ten times, 3) set the number of hidden neurons to 1000 for the synthetic dataset and UCI datasets, and 2000 for image datasets, 4) selected the regularization parameter from [0.0001, 0.001, . . ., 10000].

• In the adaptive graph learning algorithms, the two graph updating schemes are updating all weights and updating only those associated with the neighbors. The experiment 1) optimized the graph updating scheme for CAN, CLR, and PCAN, 2) reported the better performance of CLR between the L1 and Frobenius norm settings, 3) fixed the graph updating scheme (to updating all weights) for ELM-CLR to reduce the number of the user-specified parameters, 4) used the early stopping criterion of 30 iterations, 5) reported zero for ACC if the clustering solution had wrong number of clusters.

• In the fixed graph based clustering algorithms, i.e., SC, NCut, and US-ELM, the experiment constructed the graph using a self-tuning Gaussian method [16, 41].

• In all graph based algorithms, the experiment selected the numbers of neighbors from [3,. . ., 10].

• All datasets were normalized using min-max normalization method to $[-1, 1]$.

4.3.2 Clustering Accuracy on Synthetic Datasets

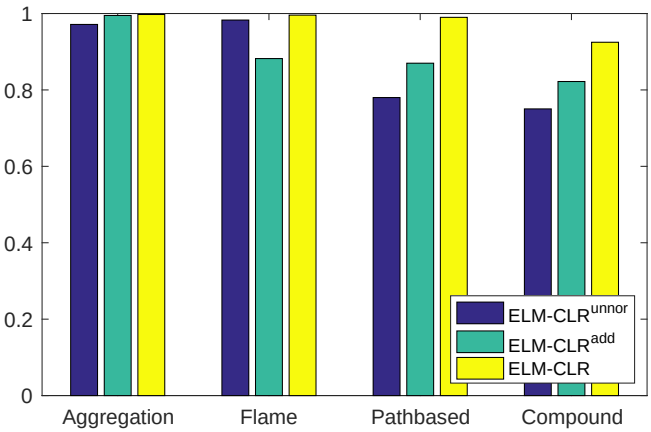


Figure 4.4: ACC (%) of ELM-CLR and its variants on the synthetic datasets.

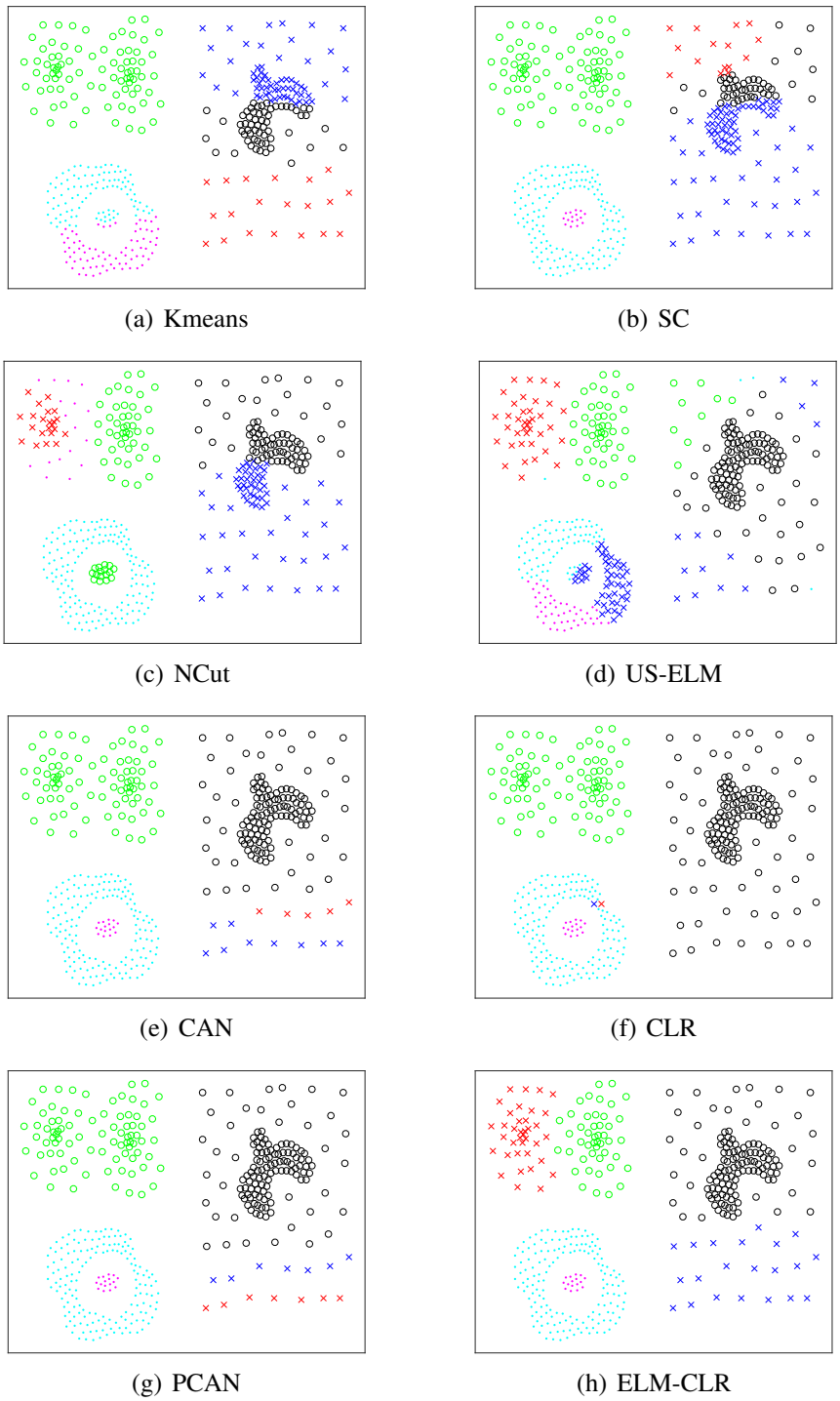
ELM-CLR and its variants were first compared on the four synthetic datasets. The clustering results are reported in Fig. 4.4. The superior performance of ELM-CLR over ELM-CLR$^{\text{unnor}}$ shows that using the normalized pairwise distances to better reveal the underlying cluster structure is beneficial for adaptive graph learning. ELM-CLR also outperformed ELM-CLR$^{\text{add}}$ on most datasets. The reason might be due to that the addition operation in ELM-CLR$^{\text{add}}$ makes the learned weights sensitive to the scale difference between distances in the original space and distances in the embedded space, which is irrelevant to understanding the underlying structure of the data.

Table 4.2: ACC (%) of ELM-CLR and the related algorithms on the synthetic datasets

	Aggregation	Flame	Pathbased	Compound
k-means	75.98 ± 5.93	84.83 ± 0.99	74.35 ± 0.07	62.36 ± 10.57
SC	88.21 ± 8.84	99.17 ± 0	85.25 ± 5.10	77.47 ± 4.72
NCut	89.68 ± 8.35	98.75 ± 0	98.00 ± 0	79.70 ± 8.10
US-ELM	76.76 ± 4.13	85.38 ± 1.67	88.23 ± 6.81	60.85 ± 5.91
CAN	99.62	98.33	87.00	80.20
CLR	**99.75**	98.33	**99.00**	77.44
PCAN	99.62	98.33	87.00	79.70
ELM-CLR	$\mathbf{99.75 \pm 0}$	$\mathbf{99.58 \pm 0}$	$\mathbf{99.00 \pm 0}$	$\mathbf{92.48 \pm 0}$

Table 4.2 presents the clustering accuracy on the synthetic datasets. ELM-CLR obtained better or comparable clustering performance compared with all the related algorithms. With the increasing enclosure of circular clusters in datasets (from Aggregation, Flame, Pathbased to Compound), the clustering solutions by adaptive graph learning algorithms (i.e., CAN, CLR, and PCAN) generally became less satisfactory. Figure 4.5 presents the visualized results on a challenging task, i.e., Compound dataset. ELM-CLR correctly clustered four out of the six clusters but mixed up the other two, one of which completely encloses the other, while the other algorithms only correctly clustered up to two.

To study how the embedding and the graph change over iterations, the study continued to record some intermediate results and visualized them in Figure 4.6 based on a small



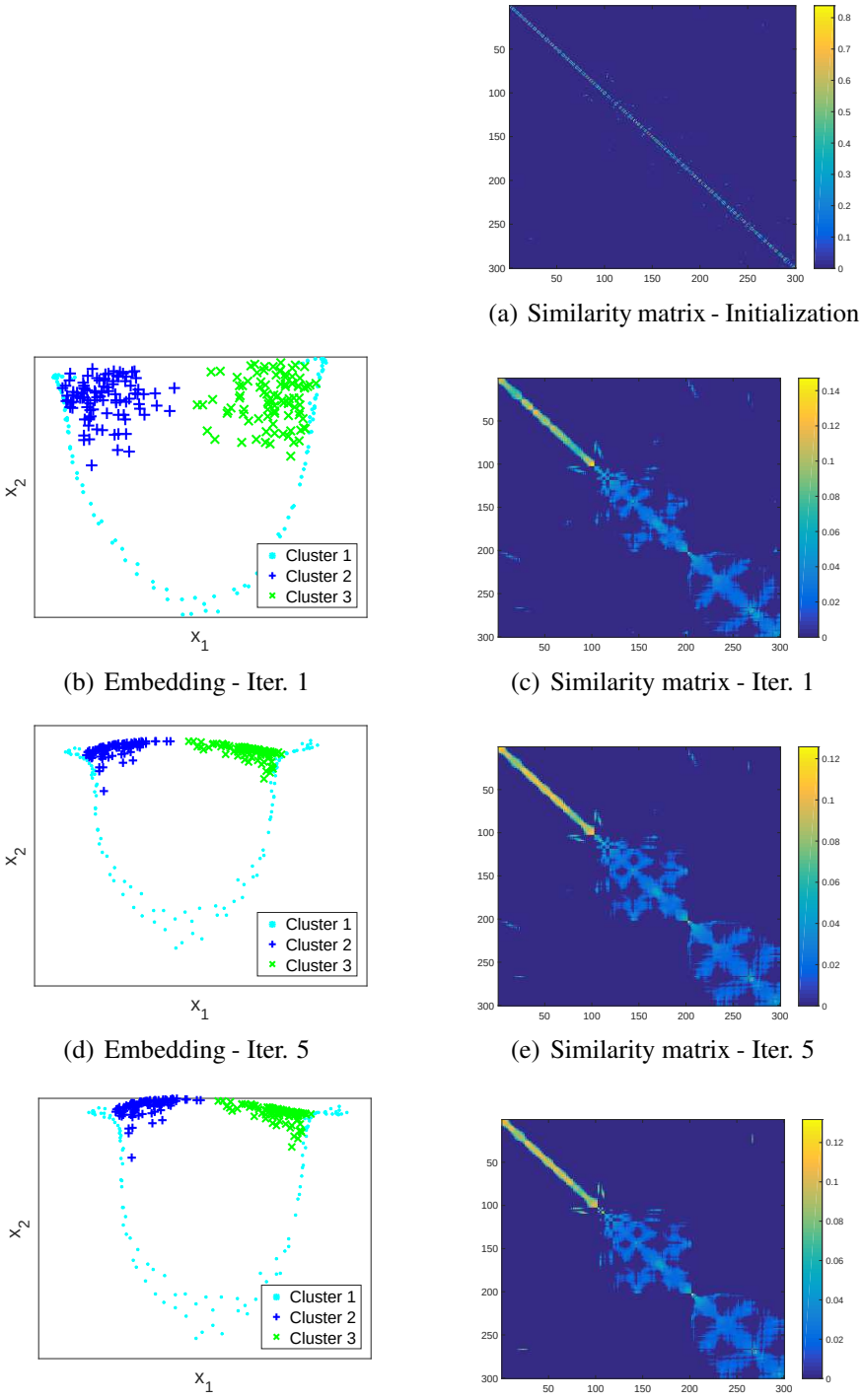
(a) Kmeans (b) SC

(c) NCut (d) US-ELM

(e) CAN (f) CLR

(g) PCAN (h) ELM-CLR

Figure 4.5: Visualization of clustering results by different algorithms on the Compound dataset. (The color coding illustrates the clustering solution by an algorithm.)



(a) Similarity matrix - Initialization

(b) Embedding - Iter. 1

(c) Similarity matrix - Iter. 1

(d) Embedding - Iter. 5

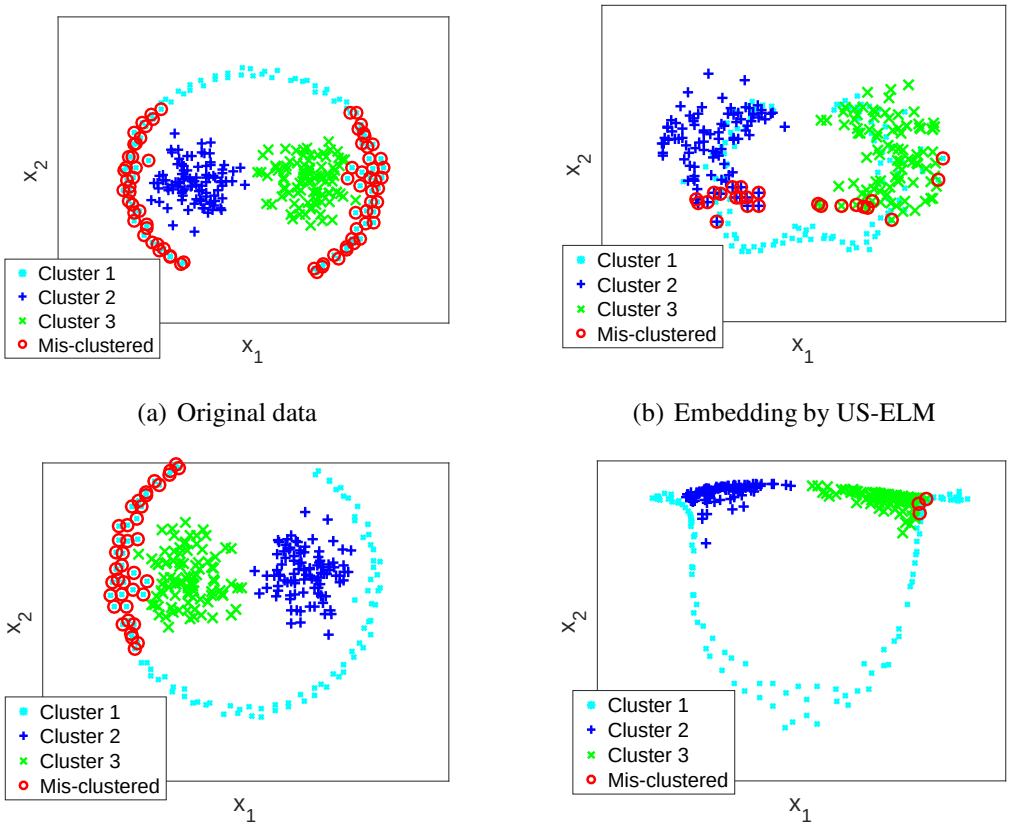
(e) Similarity matrix - Iter. 5

(f) Embedding - Iter. 10

(g) Similarity matrix - Iter. 10

Figure 4.6: Visualization of embedding and similarity matrix by ELM-CLR over different iterations on the Pathbased dataset. (On the left side are figures showing the first two dimensions of the embedding. On the right side are the figures showing the heat map plot of the graph weights. The axes represent the indexes of data points, which are ordered such that the first 100 points are of Cluster 1, the next 100 points are of Cluster 2, and the last 100 points are of Cluster 3. The color coding illustrates the similarity between two data points.)

yet challenging dataset, i.e., Pathbased. In the initialization stage, the similarity matrix formed no clear cluster structures as shown in Figure 4.6(a). In the first iteration as shown in Figure 4.6(c), some data points from Cluster 1 were wrongly connected with data points from Cluster 2 and 3. Over the ten iterations, the number of wrongly connected data points decreased as shown in the similarity matrix plot; the separation between Cluster 1 and the other two clusters increased in the embedding; the opening of the circular Cluster 1 increased in the embedding. It is worth noting that the wrong connections between Cluster 1 and Cluster 2 were eliminated in the last iteration as shown in Figure 4.6(g). Therefore, it is concluded that both the embedding and similarity matrix improve over iterations.



(a) Original data

(b) Embedding by US-ELM

(c) Embedding by PCAN

(d) Embedding by ELM-CLR

Figure 4.7: Visualization of the original Pathbased data and the embedding by US-ELM, PCAN, and ELM-CLR. (Axes represent the first two dimensions. The original data are clustered using k-means, and the mis-clustered data points are plotted using red circles. Similarly, the mis-clustered data points by US-ELM, PCAN, and ELM-CLR are indicated using red circles in the corresponding figures.)

To investigate the effects of the adaptive graph learning, the study continued to compare ELM-CLR with US-ELM, which also obtains a nonlinear embedding but relies on a fixed graph; to investigate the effects of nonlinearity, the study compared ELM-CLR with PCAN, which is an adaptive graph learning method based on linearly projected data. Figure 4.7 shows the visualization of original Pathbased data and embeddings obtained by various algorithms along with their clustering results. As shown in Figure 4.7(a), in the original space, data of Cluster 1 formed a circular shape with a small opening; k-means mis-clustered some data points of Cluster 1 that are close to Cluster 2 and 3.

The algorithms were first analyzed in terms of their effectiveness of embedding: Compared with US-ELM as shown in Figure 4.7(b), ELM-CLR produced an embedding with much less overlap among different clusters as in Figure 4.7(d). Compared with PCAN as shown in Figure 4.7(c), which more or less retained the original cluster structures, ELM-CLR effectively increased the opening of the circular Cluster 1. Intuitively, based on the visualization of the first two dimensions, the embedding by ELM-CLR was more suitable for the clustering task. Then the algorithms were analyzed in terms of their clustering performance: Both US-ELM and PCAN reduced the number of mis-clustered data points compared with k-means on the original data. As shown in Figure 4.7(d), ELM-CLR significantly outperformed US-ELM and PCAN on this problem with only three data points mis-clustered. This example demonstrates that it is beneficial to use ELM for embedding and CLR for graph learning jointly.

4.3.3 Clustering Accuracy on UCI and Image Datasets

Table 4.3 shows ACC of ELM-CLR and the related algorithms on the UCI datasets. Overall, ELM-CLR achieved better or comparable results than the related algorithms. On Iris and Ecoli datasets, ELM-CLR yielded much better clustering results than the fixed-graph counterparts, i.e., US-ELM, again showing the benefits of adaptive graph learning. As seen from the small difference between ELM-CLR and PCAN, the advan-

Table 4.3: ACC (%) of ELM-CLR and the related algorithms on the UCI datasets

Data Set	Iris	Wine	Glass	Ecoli
k-means	85.98 ± 7.86	94.68 ± 0.52	45.02 ± 3.43	57.82 ± 6.11
SC	93.18 ± 4.69	97.19 ± 0	44.37 ± 2.84	66.40 ± 4.26
NCut	93.63 ± 5.79	97.22 ± 0.12	48.23 ± 1.76	62.98 ± 4.91
US-ELM	86.15 ± 6.87	97.19 ± 0	49.04 ± 2.69	67.60 ± 4.10
CAN	88.67	94.38	45.79	81.25
CLR	**96.00**	96.07	47.66	81.25
PCAN	**96.00**	**98.88**	50.00	81.55
ELM-CLR	$\mathbf{96.00 \pm 0}$	98.71 ± 0.27	$\mathbf{50.93 \pm 0}$	$\mathbf{83.04 \pm 0}$

tage of using nonlinear embedding is not prominent, which could be because these UCI datasets are of lower than 20 dimensions and are less likely to form nonlinear structures in their original spaces.

Table 4.4: ACC (%) of ELM-CLR and the related algorithms on the image datasets

Data Set	YaleB	ORL	UMIST	FEI	COIL20	USPST	ETH-80	MNIST05
k-means	41.92 ± 3.18	49.38 ± 3.11	43.28 ± 2.34	44.39 ± 2.34	55.98 ± 4.78	65.33 ± 3.14	45.86 ± 1.99	53.22 ± 2.82
SC	48.95 ± 2.84	63.23 ± 2.41	67.91 ± 3.19	62.12 ± 2.03	77.36 ± 4.31	75.01 ± 6.32	53.25 ± 1.89	67.02 ± 5.64
NCut	49.24 ± 2.95	63.08 ± 2.13	60.66 ± 3.93	60.67 ± 2.18	71.87 ± 5.25	72.33 ± 6.10	46.79 ± 2.13	64.17 ± 3.95
US-ELM	51.63 ± 3.46	66.27 ± 2.71	65.94 ± 3.84	61.56 ± 2.30	74.68 ± 5.46	77.36 ± 5.90	53.38 ± 3.22	64.62 ± 5.23
CAN	42.42	59.50	**77.57**	58.57	90.14	74.24	17.56	58.25
CLR	46.06	61.75	75.65	65.43	87.64	77.98	27.01	**67.98**
PCAN	42.42	59.00	69.22	61.71	81.74	68.71	46.86	55.16
ELM-CLR	$\mathbf{55.94 \pm 1.03}$	$\mathbf{68.25 \pm 0}$	76.45 ± 0.34	$\mathbf{66.13 \pm 0.08}$	$\mathbf{99.72 \pm 0}$	$\mathbf{88.56 \pm 3.93}$	$\mathbf{56.77 \pm 1.23}$	67.90 ± 4.98

Table 4.4 shows the clustering performance in terms of ACC on the image clustering datasets. On most of the datasets, ELM-CLR showed the highest or the second highest clustering performance. On COIL20 and USPTS datasets, it significantly outperformed all the other algorithms by a large margin.

Although CAN and CLR showed slightly better performance than ELM-CLR on UMIST and MNIST05 datasets, they achieved very unsatisfactory results on ETH-80 dataset. There are eight generic classes in this dataset, and each class encompasses images of ten different objects. Because of the high variety of objects, data tend to form more clusters than the number of generic classes. It was difficult for CAN and CLR to construct a graph with a smaller number of connected components when the original data formed much larger number of clusters. Even linearly projection could not effectively

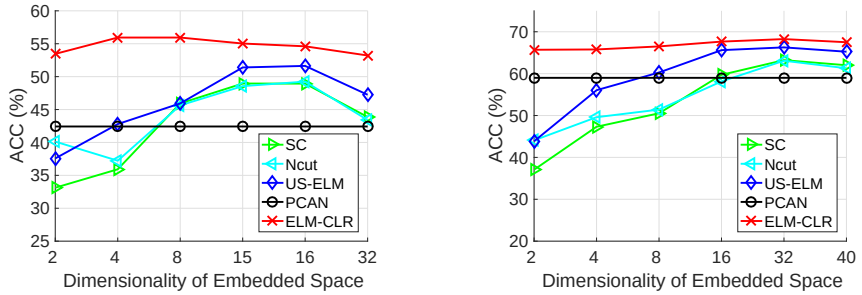
reduce clusters in the projected space, which can be seen from the performance of PCAN (close to the baseline algorithm k-means). On the other hand, ELM-CLR handled this challenge by introducing nonlinear embedding into graph learning. As long as a pair of data points are considered close in one representation (either the original representation or the nonlinear embedding), they have a higher chance of being grouped into one cluster. Based on its higher clustering performance, it is concluded that ELM-CLR can recover the generic classification better than other adaptive graph learning algorithms.

4.3.4 Influence of Dimensionality of Embedded Space

The study continued to investigate the influence of the dimensionality of embedded space on the clustering performance. Figure. 4.8 plots the clustering accuracy against the dimensionality. SC, NCut, and US-ELM were more sensitive to the dimensionality of embedded space. Such high sensitivity could be because k-means is used to extract the cluster indicators as the final step in these three algorithms; the result of k-means is sensitive to the data dimensionality and structures. On the contrary, the adaptive graph learning algorithms, i.e., PCAN and ELM-CLR, were less sensitive to dimensionality. The possible reason is that the clustering results are obtained from the graph, and the dimensionality of embedded space only indirectly affects the final clustering results via the graph. Moreover, ELM-CLR showed higher accuracy than PCAN regardless of the dimensionality of embedded space except when the dimensionality was two on MNIST05 dataset.

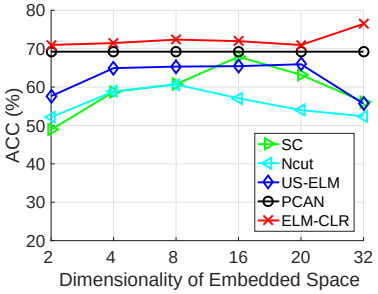
4.3.5 Influence of Trade-off Parameters

The trade-off parameter during ELM network learning step (i.e., in Equation (4.15)) is δ. The trade-off parameter γ during graph initialization and graph learning (i.e., in Equation (4.13) and Equation (4.7), respectively) is calculated based on the desired number of neighors k, which is specified by the user. Under the optimized the dimensionality of

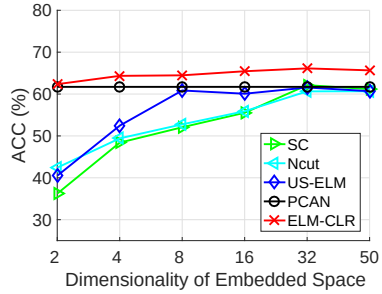


(a) YaleB

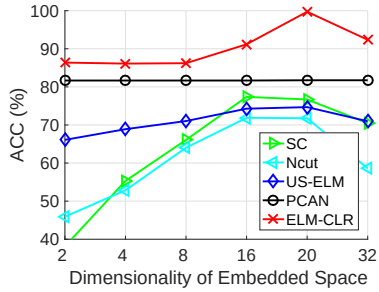
(b) ORL



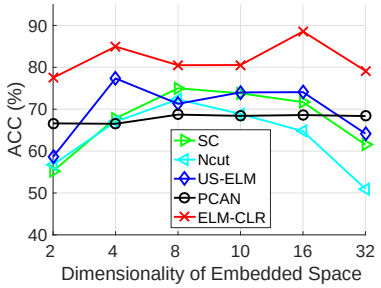
(c) UMIST



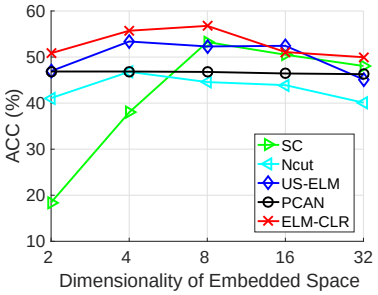
(d) FEI



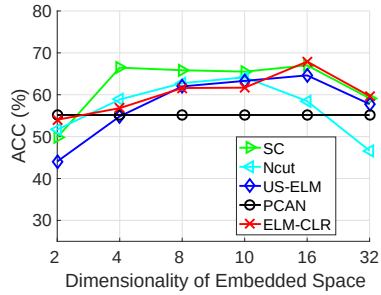
(e) COIL20



(f) USPST

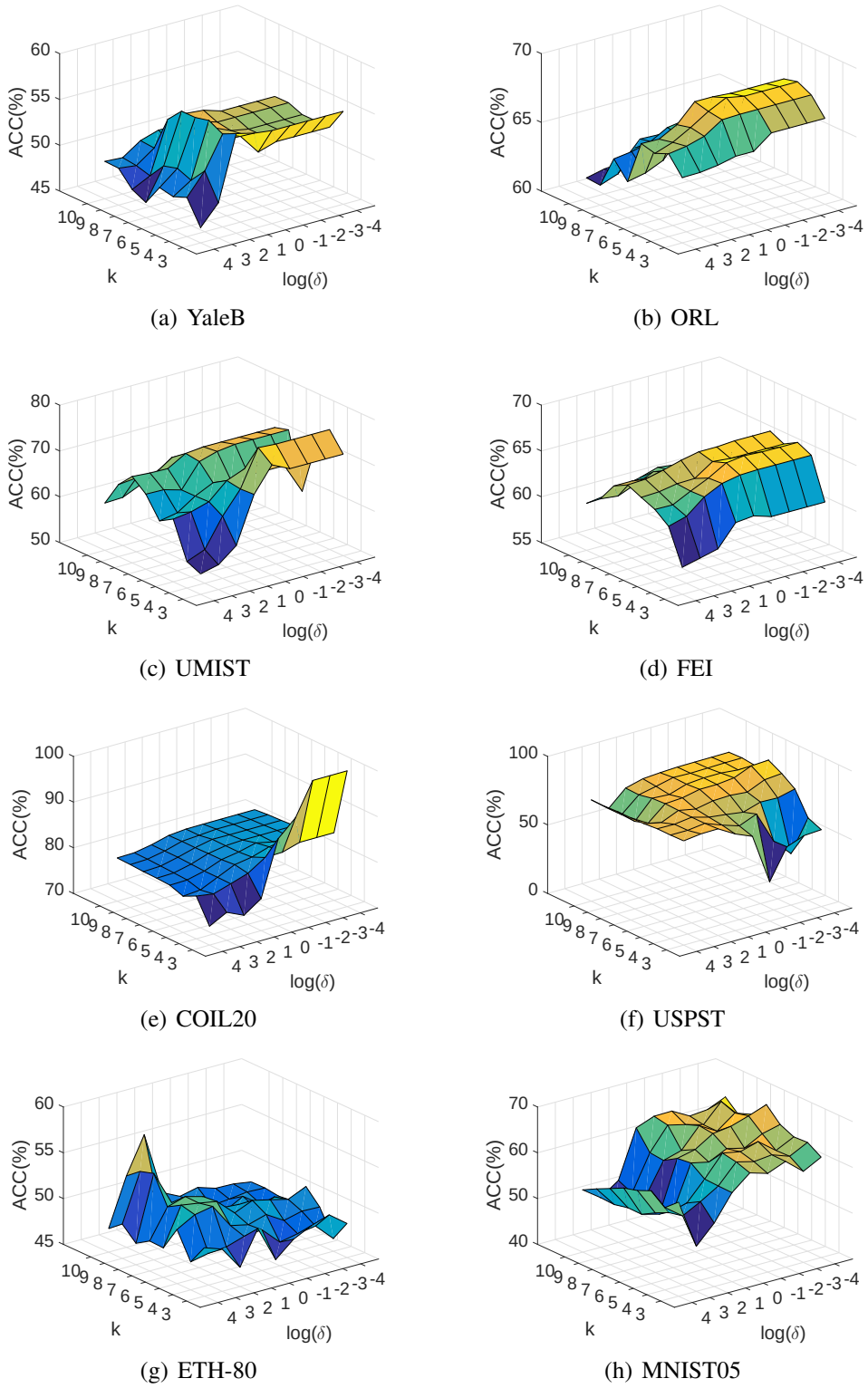


(g) ETH-80



(h) MNIST05

Figure 4.8: ACC (%) of SC, NCut, PCAN, US-ELM, and ELM-CLR with different dimensionalities of embedded spaces.



(a) YaleB (b) ORL

(c) UMIST (d) FEI

(e) COIL20 (f) USPST

(g) ETH-80 (h) MNIST05

Figure 4.9: ACC(%) of ELM-CLR with different values of regularization parameter δ and the number of neighbors k.

embedded space, the plots of the performance of ELM-CLR using different values of δ and k are shown in Fig. 4.9.

As shown in Fig. 4.9, the performance of ELM-CLR on most datasets was quite smooth with respect to the trade-off parameter values. Therefore, it is possible to select the trade-off parameters using similar datasets with labels. For instance, the optimal parameter values of the digit clustering dataset USPST also leads to reasonable results on the other digit clustering problem MNIST05. By inspecting the sample images in Fig. 4.3, the first four face clustering datasets look similar. Similarly as shown in the face image clustering results in Fig. 4.9(a) to Fig. 4.9(d), it is safe to assign a small value to δ, e.g., 10^{-3}, which means the consistency with the graph plays the major role during the ELM network learning, and a larger value of k, which generates a more connected initial graph. Nonetheless, in order to achieve the best performance, a finer parameter selection is required, especially on datasets like ETH-80.

4.3.6 Computational Complexity

Table 4.5: The number of iterations of ELM-CLR and adaptive graph learning methods

Data Set	YaleB	ORL	UMIST	FEI	COIL20	USPST	ETH-80	MNIST05
CAN	9	8	5	11	10	16	10	16
CLR	13	11	28	13	22	16	16	21
PCAN	5	6	6	6	6	6	6	7
ELM-CLR	5	6	4	5	5	5	9	12

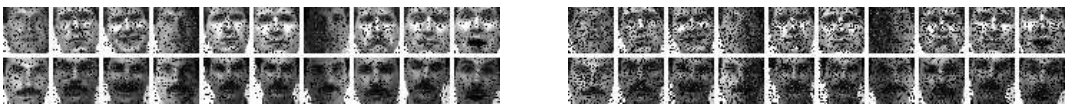
To investigate the computational complexity, all the algorithms were repeated for ten times in Matlab environment on a PC with 32GB RAM and an Intel Core i7-4790 CPU running at 3.60GHz. Table 4.5 reports the averaged number of iterations over ten runs by all the adaptive graph learning methods. PCAN and ELM-CLR took a smaller number of iterations than CAN and CLR on most datasets. This shows that updating the graph based on the original features, as done by CAN and CLR, is not as effective as on some projected features, in terms of convergence speed towards a graph with the desired number of connected components.

Table 4.6: The computational time (in seconds) of ELM-CLR and the related algorithms on the image datasets

Data Set	YaleB	ORL	UMIST	FEI	COIL20	USPST	ETH-80	MNIST05
k-means	0.008	0.033	0.026	0.064	0.214	0.075	0.603	0.703
SC	0.009	0.200	0.207	0.098	0.393	1.078	4.621	5.442
NCut	0.014	0.046	0.052	0.098	0.205	0.258	0.799	0.857
US-ELM	0.056	0.136	0.128	0.231	0.929	1.229	1.914	2.514
CAN	0.041	0.198	0.359	0.720	4.735	16.522	45.392	82.455
CLR	0.064	0.370	1.198	0.911	7.402	22.780	88.930	130.414
PCAN	5.601	8.108	1.892	3.611	7.796	15.483	45.299	50.127
ELM-CLR	0.400	1.450	1.857	2.635	11.770	18.391	102.656	158.867

The overall computational time of ELM-CLR and other methods were also compared in Table 4.6. The first four algorithms do not adaptively learn the graph and thus took less computational time than the adaptive graph learning methods. ELM-CLR took longer time than the other adaptive graph learning methods especially on datasets with large size, yet only up to 1.5 times of the longest time among CAN, CLR, and PCAN. This shows that using ELM to achieve the nonlinear embedding is an efficient choice.

4.3.7 Robustness against Noise



(a) 15% of the original pixels are replaced (b) 30% of the original pixels are replaced

Figure 4.10: Sample images from YaleB dataset with 15% and 30% of original pixels replaced by random values following uniform distribution between -0.5 and 0.5.

Noise is a common challenge in real world applications. To study the robustness against noise, a series of corrupted datasets were created based on the YaleB dataset with different levels of noise and are shown in Figure 4.10. The clustering accuracy is plotted against the level of noise in Figure 4.11. All algorithms showed degrading performance with increasing level of noise. ELM-CLR was slightly more robust than the other al-

gorithms, as the performance of ELM-CLR only decreased by less than 5% when the noise level was 15%. When the noise level was more than 15%, all algorithms degraded significantly. How to handle a large amount of noise is an important future work.

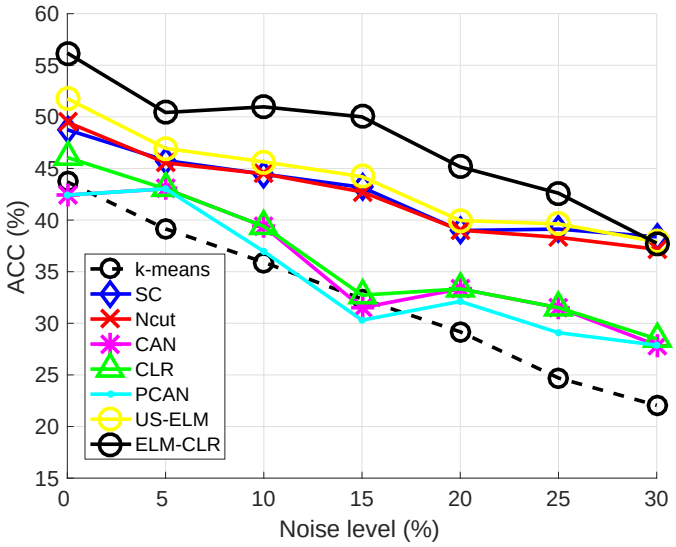


Figure 4.11: ACC (%) on the corrupted YaleB dataset with different levels of noise.

4.3.8 Influence of Different Cluster Numbers

We investigate how the proposed algorithm handle the situation where the number of cluster is specified to be a smaller value. This is useful in the cases where our desired number of clusters is smaller than the number of true clusters formed in the data.

On COIL20 dataset, the 20 clusters are shown in Fig. 4.12. The clustering results by ELM-CLR with 19 clusters and 17 clusters are shown in Fig. 4.13 and Fig. 4.14, respectively.

We observe that every time we reduce the number of classes, ELM-CLR merges two similar clusters together. The 2nd (the arrow shape object) and 8th (the milk bottle) clusters in Fig. 4.12 are merged as the 2nd cluster in Fig. 4.13. The 3rd, 6th and 7th clusters in Fig. 4.13 are all toy cars and are merged as the 3rd cluster in Fig. 4.14. Based on human intuition and observation, the similar clusters grouped by ELM-CLR

are sensible. For example, indeed the three clusters of three toy cars are very similar, and the arrow shape object and the milk bottle look very similar from some angle.

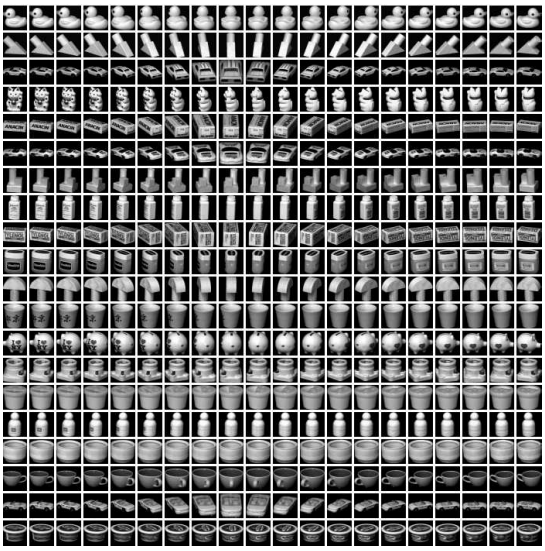


Figure 4.12: The 20 classes of the COIL20 dataset.

4.4 Summary

In this chapter, the first research objective of the thesis is addressed, i.e., to develop effective clustering algorithms by taking advantage of ELM. In Section 4.1, a popular clustering framework, i.e., adaptive graph learning for clustering, is reviewed. To address its limitations, Section 4.2 proposes an adaptive graph learning method, ELM-CLR, to further improve the clustering performance and robustness by adjusting the graph based on dual data representations. ELM-CLR retains the advantages of other adaptive graph learning algorithms, such as not requiring other clustering methods to extract cluster indicators from the graph. Moreover, ELM-CLR is the first to introduce nonlinear embedding into adaptive graph learning. As a result, it can discover generic clusters and demonstrates low sensitivity to the dimensionality of embedded space and noise. As shown in Section 4.3, on a board range of datasets, ELM-CLR achieved more favorable clustering performance than the related state-of-the-art algorithms.

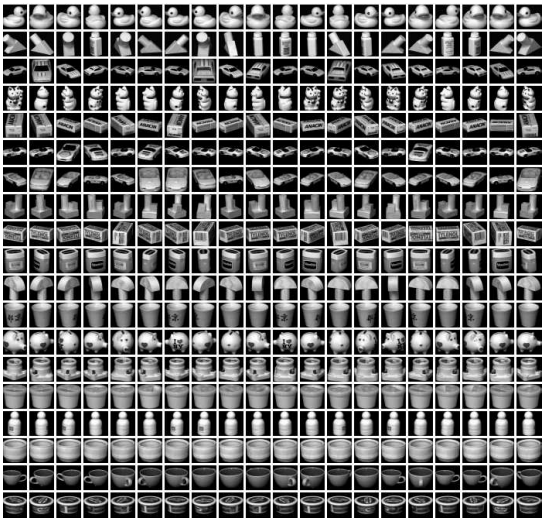


Figure 4.13: Clustering results given by ELM-CLR with 17 clusters (ELM-CLR cannot find the result with cluster number of 18)

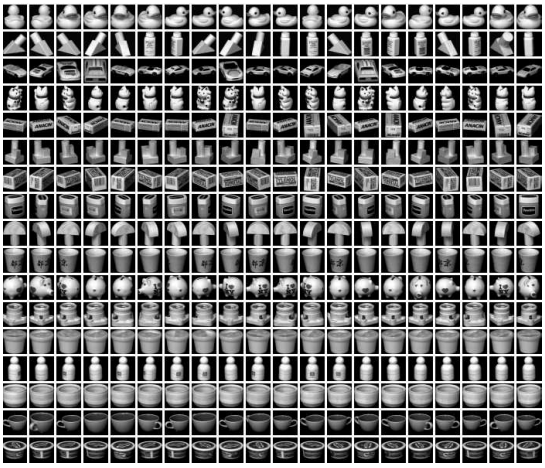


Figure 4.14: Clustering results given by ELM-CLR with 17 clusters (ELM-CLR cannot find the result with cluster number of 18)

Chapter 5

Application to Driver Distraction Detection

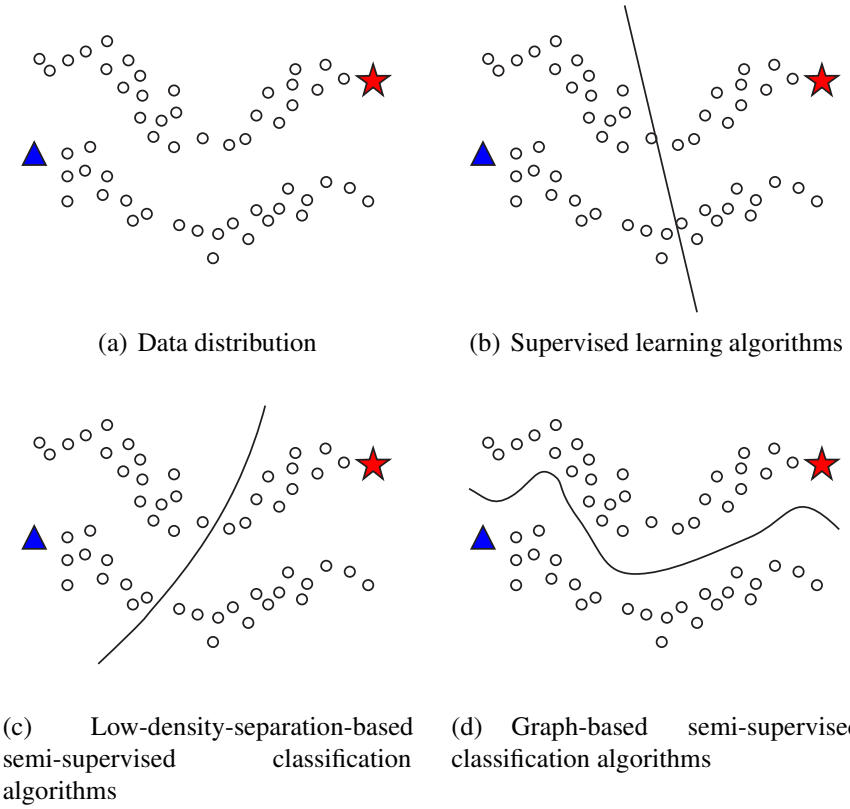
Chapter 5 empirically studies the application of clustering and semi-supervised classification to driver distraction detection problem. Firstly, the background and motivation is presented in Section 5.1, followed by the proposed semi-supervised driver distraction detection system with labeling assistance tool in Section 5.2. Then the data set is described in Section 5.3. Section 5.4 reports the experiments and results of the semi-supervised detection system. Section 5.5 reports the experiments and results of clustering-based labeling assistance.

5.1 Background and Motivation

Traditional machine-learning-based detection models [8, 9, 10, 55, 56, 59] rely on fully-labeled training data, which has several drawbacks, such as high safety risk, low user acceptance, and proneness to labeling errors. Firstly, the collection of positive labeled training data requires the human driver to drive with secondary task/distractions, which is highly unsafe. Secondly, before a driver can use the assistance system, collecting high-quality labeled training data under the cooperation of the driver for a long time harms the user experience. Lastly, if there are not enough information sources to assist the labeling (such as face video recording), the labels are erroneous. On the other hand, it is relatively easy to collect data without labels (distraction states), e.g., from the driver's naturalistic driving records. Therefore, for applications like driver distraction detection, it is quite necessary to 1) make better use of unlabeled training data and 2) reduce the labeling cost.

Semi-supervised classifcation algorithms can make use of both labeled and unlabeled training data. Figure 6.1 illustrates the decision boundaries given by different types of semi-supervised classification algorithms. The triangle and the star represent the labeled data points from two classes respectively, while the unlabeled data points are represented by circles. Supervised learning algorithms are trained using only labeled data and may generate a decision boundary as shown in Figure 5.1(b). By utilizing the unlabeled data, both types of semi-supervised classification algorithms could adjust the decision boundaries according to their assumptions and could give significantly different decision boundaries as shown in Figure 5.1(c) and Figure 5.1(d). Consequently, if there is a mismatch between the underlying assumption and the data distribution, the semi-supervised classification algorithm might result in even worse performance than the supervised learning algorithms. Several theoretical works have tried to address this limitation by adjusting the impact of unlabeled data within the clustering assumption framework [72]. However, there is not yet an explicit approach to select the appropriate semi-supervised classification algorithm given a new problem. Therefore, it is an

open research question "which type of semi-supervised classification algorithm is more suitable for driver distraction detection?"



(a) Data distribution

(b) Supervised learning algorithms

(c) Low-density-separation-based semi-supervised classification algorithms

(d) Graph-based semi-supervised classification algorithms

Figure 5.1: Illustration of decision boundaries given by different types of learning algorithms.

Graph-based semi-supervised classification methods have shown promising performance in many applications, such as text classification [73] and image classification [7]. One explanation is that features have low intrinsic dimensions corresponding to illumination, shape, and angle, etc. It is reasonable to assume that features used in driver distraction detection are results of some low dimensional movements in three-dimensional space, e.g., pitch, yaw, and roll of eye and head. To verify the hypothesis that eye and head movement features satisfy manifold assumption, this chapter systematically evaluates two graph-based semi-supervised classification algorithms in comparison with supervised learning algorithms and low-density-separation-based semi-supervised classification algorithms.

In addition, in order to save the effort of human experts while labeling, this study applies clustering algorithms to generate some preliminary label information. As a result, the human experts only need to confirm or correct the labels rather than creating them from scratch. This chapter also investigates the two clustering methods proposed in the previous two chapters in terms of their ability to generate accurate preliminary distraction labels.

5.2 Proposed System

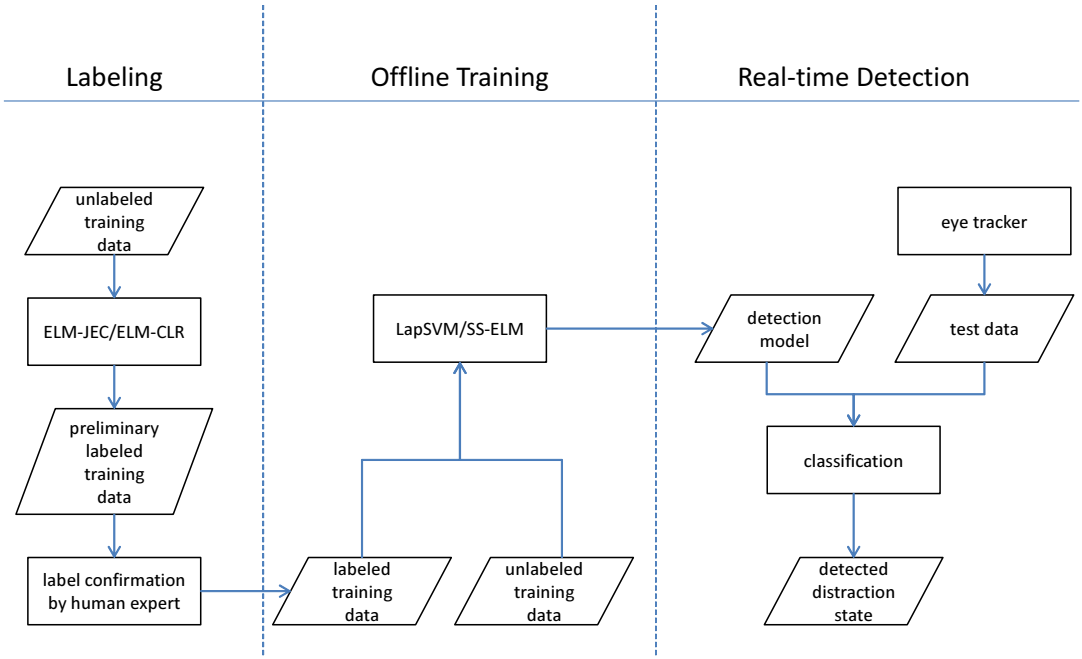


Figure 5.2: Flow chart of semi-supervised distraction detection system with labeling assistance.

Driver distraction detection is a binary classification problem including 'distracted' and 'non-distracted' classes. The flow chart of the proposed semi-supervised based distraction detection system is presented in Figure 5.2. In the labeling stage, the unlabeled training data are first passed to clustering methods, such as ELM-JEC and ELM-CLR, to obtain the preliminary labeled training data. The outputs of clustering algorithms are first checked by experts, who can focus their activity on confirming or changing

the labels, rather than creating the labels from scratch. Subsequently, the final accurate label set is used as the final labels. In the offline training stage, the graph-based semi-supervised classification algorithms, i.e., LapSVM and SS-ELM, generate a detection model. In the detection stage, test data collected from an eye tracker in real time are classified into one of the driver states according to the detection model. Finally, the complete system provides the detected distraction state as the output. Since ELM-JEC and ELM-CLR have been introduced in Chapter 3 and Chapter 4, the following two sections briefly introduce the graph-based semi-supervised classification algorithms.

5.2.1 Laplacian Support Vector Machine

Laplacian Support Vector Machine (LapSVM) [36] is a typical graph-based semi-supervised learning algorithm. Similar to its origin, Support Vector Machines (SVMs) [74], LapSVM aims to find a linear decision boundary in a higher-dimensional Hilbert space that separates the two classes of training data with maximum margin. Given a data point $\boldsymbol{x}$, the decision function by LapSVM is expressed as $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}) + b$, where $\boldsymbol{\phi}(\cdot)$ is a nonlinear mapping function, and $\boldsymbol{w}$ and b are parameters of the decision boundary to be learned. In addition, LapSVM makes use of unlabeled training data under the manifold regularization framework. The predicted binary label is finally obtained by $\text{sign}(f(\boldsymbol{x}))$.

In semi-supervised learning, few labeled data are a large amount of unlabeled data are available. The labeled data are denoted as $\{\mathcal{X}_l, \mathcal{T}_l\} = \{\boldsymbol{x}_i, t_i\}_{i=1}^{l}$, and the unlabeled data are denoted as $\mathcal{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{u}$, where $t_i \in \{-1, 1\}$ is the label corresponding to the sample $\boldsymbol{x}_i$, and l and u are the numbers of labeled and unlabeled samples respectively.

The mathematical formulation of LapSVM is as follows:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma_A\|\boldsymbol{w}\|_2^2 + \frac{\gamma_I}{(l+u)^2}\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y} \\ \text{s.t.} \quad & t_i f(\boldsymbol{x}_i) \geq 1-\xi_i,\ i=1,\ldots,l, \\ & \xi_i \geq 0,\ i=1,\ldots,l, \\ & f(\boldsymbol{x}_i) = \boldsymbol{w}\cdot\boldsymbol{\phi}(\boldsymbol{x}_i) + b,\ i=1,\ldots,l+u \end{aligned} \tag{5.1}$$

where $\boldsymbol{L}$ is the graph Laplacian calculated based on the training data as in Section 2.4.1, ξ_i is the error caused by the i-th training sample, γ_A and γ_I are the trade-off coefficients, $\boldsymbol{y} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_{l+u})]^T$ is the output vector, and $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_l]^T$ is the error vector. The objective function aims to minimize the empirical training error, the inverse square of the margin, and the manifold regularization term.

By representer theorem [75], the minimizer of the above optimization problem admits an expansion as

$$f^*(\boldsymbol{x}_i) = \sum_{j=1}^{l+u}\alpha_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + b, \tag{5.2}$$

where $K(\boldsymbol{x}_j, \boldsymbol{x}_i) = \boldsymbol{\phi}(\boldsymbol{x}_j)\cdot\boldsymbol{\phi}(\boldsymbol{x}_i)$ is the kernel function which can be defined without explicitly specifying the nonlinear mapping function $\boldsymbol{\phi}(\cdot)$. In this case, $\boldsymbol{w}$ is equivalent to $\sum_{i=1}^{l+u}\alpha_i\phi(\boldsymbol{x}_i) = \boldsymbol{\phi}\boldsymbol{\alpha}$. As a result, the optimization problem can be rewritten as:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad & \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma_A\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + \frac{\gamma_I}{(l+u)^2}\boldsymbol{\alpha}\boldsymbol{K}\boldsymbol{L}\boldsymbol{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & t_i\left(\sum_{j=1}^{l+u}\alpha_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) + b\right) \geq 1-\epsilon_i, i=1,\ldots,l \\ & \epsilon_i \geq 0, i=1,\ldots,l \end{aligned} \tag{5.3}$$

where $\boldsymbol{K}$ is the gram matrix of dimension $(l+u)\times(l+u)$.

This problem can be further reformulated as a linear system and a quadratic program-

ming problem, which can be solved using a standard SVM solver [36]. Furthermore, this problem can also be solved in the primal with shorter training time [76].

5.2.2 Semi-Supervised Extreme Learning Machine

Semi-Supervsed Extreme Learning Machine (SS-ELM) [15] is an extension of the basic ELM based on manifold regularization framework. In a classification problem, given a data point $\boldsymbol{x}$, the output of SS-ELM is $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} \in \mathbb{R}^{1\times n_O}$, where $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^{1\times n_H}$ is the output vector of the hidden layer and $\boldsymbol{\beta} \in \mathbb{R}^{n_H\times n_O}$ is the weight connecting hidden layer and output layer. The predicted class is the label corresponding to the entry with the largest value in the output vector. This enables SS-ELM to handle multi-class classification naturally.

In semi-supervised learning, the labeled data are denoted as $\{\mathcal{X}_l, \mathcal{T}_l\} = \{\boldsymbol{x}_i, \boldsymbol{t}_i\}_{i=1}^{l}$ and the unlabeled data are denoted as $\mathcal{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{u}$, where l and u are the numbers of labeled and unlabeled samples respectively. Here $\boldsymbol{t}_i \in \{0,1\}^{n_O}$ is the class label vector with only the k-th entry equal to one, where k is the index of the class that $\boldsymbol{x}_i$ belongs to. Then the manifold regularization term can be expressed as:

$$L_m = \frac{1}{2}\sum_{i,j} w_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 = \operatorname{Tr}\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right), \tag{5.4}$$

where $\boldsymbol{L}$ is the graph Laplacian calculated based on the training data as in Section 2.4.1 and $\boldsymbol{Y} = [\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_n^T]^T = \boldsymbol{H}\boldsymbol{\beta} \in \mathbb{R}^{(l+u)\times n_O}$ is the output matrix of the network.

By adding the manifold regularization term to the ordinary ELM formulation, Huang et al. [15] proposed the mathematical formulation of SS-ELM as follows:

$$\min_{\boldsymbol{\beta}} \quad \frac{1}{2}\left\|\boldsymbol{C}^{\frac{1}{2}}(\tilde{\boldsymbol{T}} - \boldsymbol{H}\boldsymbol{\beta})\right\|_F^2 + \frac{1}{2}\|\boldsymbol{\beta}\|_F^2 + \frac{\lambda}{2}\operatorname{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{L}\boldsymbol{H}\boldsymbol{\beta}\right) \tag{5.5}$$

where $\tilde{\boldsymbol{T}} \in \mathbb{R}^{(l+u)\times n_O}$ is the training target with its first l rows equal to $[t_1, t_2, \ldots, t_l]^T \in \mathbb{R}^{l\times n_O}$ and the rest equal to 0, $\lambda > 0$ is the trade-off coefficient for manifold regulariza-

tion. $\boldsymbol{C}$ is a diagonal matrix with $c_{ii} = \frac{C}{n_{t_i}}$, where n_{t_i} is size of the class that the i-th training sample belongs to.

The problem can be solved in a closed form as follows:

$$\begin{aligned} \boldsymbol{\beta} &= \left(\boldsymbol{I}_{n_H} + \boldsymbol{H}^T\boldsymbol{C}\boldsymbol{H} + \lambda\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{C}\tilde{\boldsymbol{T}}, \; n_H \leq (l+u), \\ \boldsymbol{\beta} &= \boldsymbol{H}^T\left(\boldsymbol{I}_{l+u} + \boldsymbol{C}\boldsymbol{H}\boldsymbol{H}^T + \lambda\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{C}\tilde{\boldsymbol{T}}, \text{ otherwise.} \end{aligned} \tag{5.6}$$

In summary, SS-ELM obtains the model in three steps: 1) generate the parameters of the hidden layer randomly; 2) compute the output matrix of the hidden layer; 3) solve the output weight using (5.6).

5.3 Description of Data Set

The proposed semi-supervised driver distraction detection system is evaluated using a real-world dataset of eye and head movements collected in the UK by a team lead by Dr. Yang [77]. In brief, the participants performed a driving task on a pre-defined route in the inner lane of a dual-carriageway trunk road using an instrumented vehicle; in addition to the driving task, they were occasionally asked to perform some secondary tasks which simulated distractions.

5.3.1 Subjects

Experience drivers aged from 21 to 65 were recruited through different channels including email advertisements, word-of-mouth, and leafleting. 29 males and 12 females drivers jointed the experiment. Finally, seven subjects were excluded due to some technical problems and incomplete data. The final dataset used in this study consisted of records from 23 males and 11 females.

5.3.2 Apparatus

An instrumented vehicle was used in the experiment, including a Fiat Stilo motor car, as shown in Figure 5.3(a), and sensors to measure vehicle dynamics, eye and head movements, and other behavioral actions. As shown in Figure 5.3(b), main apparatus for data collection and experiment monitoring include faceLAB eye tracker (Item 1), the laptop used to faceLAB eye tracking software (Item 2), and the laptop displaying real-time eye gaze overlaid onto the scene ahead (Item 3), respectively. The faceLAB eye traker consisted of two infrared cameras. Before the data collection experiment started, the faceLAB system was calibrated for each driver to account for their eye and facial features. The calibration took around 15 mins. The data quality was ensured by running all experiments without rain and direct sun light.

(a) Instrumented vehicle (b) Apparatus layout

Figure 5.3: Photos of the on-road experiment for data collection.

5.3.3 Tasks and Procedures

The primary task given to the driver was a car-following task. Another car moved on the same road slightly ahead of the instrumented car. The driver needed to maintain a consistent head-way with the leading car by focusing on the road ahead. Occasionally, the driver was also instructed to perform a "sound counting" task simulating cognitive distraction without requiring too much visual attention. Specifically, some specific sounds were given to the driver at the beginning of the task and were called the "target sounds"

such as a vehicle honking, and the driver needed to remember the target sounds. Later on, after hearing a subsequent auditory clip, which consisted of a mixture of target and non-target sounds, e.g., phone rings, the driver needed to count the number of target sounds. All the sounds lasted about 320 ms and were played with a gap of two seconds.

Each set of secondary task lasted around 50 seconds, creating one *section of driving with secondary tasks* (mean = 52.9 seconds, st = 7.8 seconds). For each driver, nine sets of secondary task were repeated with a gap of 15 seconds (for the driver to recover from distracted state before the next section started). *Sections of normal driving* were also conducted for 73 to 279 seconds depending on how much road were left before the route was completed. The drivers were given sufficient time to practise both the driving task and the secondary tasks in order to rule out any learning effects. All data collection experiments started at either 9:30 or 14:00 to avoid peak hour traffic.

5.3.4 Eye and Head Movement Measures

Eye and head movement data were collected at 60 Hz. A list of measures, as shown in Table 5.1, is identified from outputs of the eye tracking software [78] and is considered beneficial for cognitive distraction detection. Specifically, the head position is measured relative to World Coordinate, where the origin is fixed at the center point between the two infrared cameras. The gaze rotation is defined as the vector pointing from the center of eye ball towards the object being looked at.

As defined in Section 2.5.1, some eye-movements are also detected and provided by the eye tracking software. A saccade is a rapid eye movement used in repositioning the fovea to a new location in the visual environment [79]. The saccade measure in this study is a binary value indicating whether there is a saccade or not. A blink is defined as a rapid eye closure followed quickly by a rapid eye opening. Similarly, the blink measure is a binary value indicating whether there is a blink event or not. The blink frequency and blink duration were calculated based on a sliding window of 60

seconds. The percentage of eye closure (PERCLOS) [80] is the percentage of extended periods with practically closed eyes in a time window (with parameters of a threshold $t_p(= 0.75$ by default) of eye closure and the window size $w(= 3$ minutes by default)).

Table 5.1: Details of eye and head movement measures

Group	Measure	Unit
Head Position (HP)	x	meters
	y	meters
	z	meters
Head Rotation (HR)	Yaw	radians
	Roll	radians
	Pitch	radians
Gaze Rotation (GR)	Right - Yaw	radians
	Right - Pitch	radians
	Left - Yaw	radians
	Left - Pitch	radians
Gaze Temporal (GT)	Saccade	-
	Blink	-
	Blink frequency	hertz
	Blink duration	seconds
	PERCLOS	-

5.3.5 Features and Labels

Two window-wise features as defined in Section 2.5.1 were computed and used as the final features. Specifically, the statistical features, i.e., mean of all measures and standard deviation of the first ten spatial measures, were extracted from sliding windows of 10 seconds and 95% overlap. As a result, different numbers of training samples were obtained for different subjects (mean = 1053, std = 150), mainly due to individual's difference in handling secondary tasks and traffic condition difference.

The secondary task approach, as reviewed in Section 2.5.2, was used to label the train-

ing data. Data collected in *sections of driving with secondary tasks* were labeled as "distracted driving", while data collected in *sections of normal driving* were labeled as "non-distracted driving." The resulting dataset of each subject was imbalanced with the majority class being "non-distracted driving" at an imbalanced ration of 0.45 (the size of minority class over the size of majority class).

5.4 Experiments of Semi-Supervised Detection

The objective of the experiment is to investigate the performance of graph-based semi-supervised learning algorithms, i.e., LapSVM [36] and SS-ELM [15], in comparison with the state-of-the-art supervised learning algorithms, i.e., Static Bayesian Network with Supervised Clustering (SBN-SC) [55, 56], SVM [74] and ELM [20], and the low-density-separation-based semi-supervised classification algorithm, i.e., TSVM [46].

In the following sections, features and labels are first explained, the partition of training and test sets is then presented, implementation details of all the algorithms used in the experiments are recorded, and the evaluation metrics used are defined. The experimental results are presented in five aspects, namely, the feature analysis, the comparison with the related methods, the influence of unlabeled data, the influence of labeled data, and the influence of hyper-parameters. Finally, a discussion around the results is presented.

5.4.1 Training and Test Sets

Figure 5.4 shows an illustration of data partition schemes: the complete data set from one subject was first partitioned into a training set and a test set and then the training set was further partitioned into a labeled set and an unlabeled set. The partition of the complete data set into training and test set was done using four-fold cross validation, i.e., the complete data set was randomly partitioned into four non-overlapping folds, where one fold was used for testing and the rest for training. Each fold was used for testing

once and the cross validation was repeated for three times, resulting in 12 different partitions (called as *splits*) for each subject. The training set was used for constructing the detection model, and the detection performance of the model on the test set was recorded as the testing performance. An average testing performance over the 12 splits was used to evaluate the learning algorithm. In addition, the training set was also used to obtain the parameters of z-score normalization, which were then applied to test set.

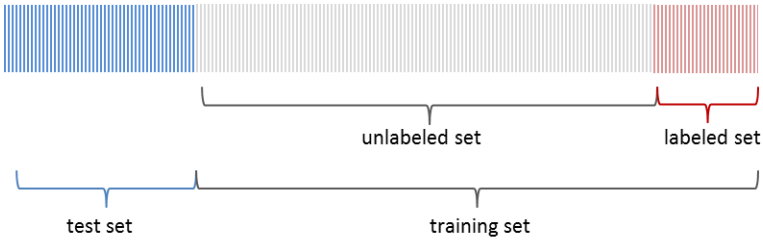


Figure 5.4: Illustration of data partition schemes.

The labeled set and the unlabeded set were formed by randomly sampling without replacement from the training set. The sizes of the labeled set and the unlabeled set were determined according to the requirement of each experiment. The labels of the unlabeled set were removed and not available during training. For a fair comparison, the data partition was conducted once and saved for all the algorithms.

5.4.2 Evaluation Metrics

Two criteria are commonly used to evaluate the importance of each feature for a classification problem: the Pearson correlation coefficient and the class separability. The correlation coefficient between two variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as

$$\text{Pearson Correlation Coefficient} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{5.7}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ indicates the sample mean of variable x, and analogously for $\bar{y}$. In this study, the correlation coefficient was computed between a feature vector and the target class vector. A high correlation coefficient of a feature vector with the target class

vector indicates that the feature is important for detecting driver distraction . Pearson correlation coefficient is widely used as a criterion for supervised feature selection [81].

The class separability of the m-th feature is defined as:

$$\text{Class Separability} = \text{Tr}\left((\boldsymbol{S}_b)^{-1}\boldsymbol{S}_w\right), \tag{5.8}$$

where $\boldsymbol{S}_w \in \mathbb{R}^{(d-1)\times(d-1)}$ and $\boldsymbol{S}_b \in \mathbb{R}^{(d-1)\times(d-1)}$ are computed based on the labeled data excluding the m-th feature as defined in Section 2.3. This is essentially the inverse of the famous Fisher discriminant ratio [37]. A large value of $\text{Tr}\left((\boldsymbol{S}_b)^{-1}\boldsymbol{S}_w\right)$ of the remaining features indicates that the data are highly mixed up. Intuitively, if removing a feature makes the class more mixed up, it implies that the feature contributes a lot to separate the classes. Therefore, the larger the class separability value, the more important the feature.

In binary classification, true positive (TP), true negative (TN), false positive (FP), and false negative (FN) are defined as follows:

$$\begin{aligned}
\text{TP} &= \textstyle\sum_1^n \delta(t_i, t_i^p) \cdot \delta(t_i^p, 1), \\
\text{TN} &= \textstyle\sum_1^n \delta(t_i, t_i^p) \cdot \delta(t_i^p, 0), \\
\text{FP} &= \textstyle\sum_1^n \tilde{\delta}(t_i, t_i^p) \cdot \delta(t_i^p, 1), \\
\text{FN} &= \textstyle\sum_1^n \tilde{\delta}(t_i, t_i^p) \cdot \delta(t_i^p, 0),
\end{aligned} \tag{5.9}$$

where n is the size of the dataset, $t_i \in \{0, 1\}$ is the ground truth class label, a value of one represents the distracted driving class and a value of zero represents the normal driving class, t_i^p is the predicted class label. $\delta(a, b) = 1$ if $a = b$, otherwise, $\delta(a, b) = 0$. $\tilde{\delta}(a, b) = 1$ if $a \neq b$, otherwise, $\tilde{\delta}(a, b) = 0$.

Based on the above indicators, some performance metrics used in this study are defined,

i.e., accuracy, sensitivity, specificity, and G-mean.

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}}, \\
\text{Sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\
\text{Specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}}, \\
G\text{-mean} &= \sqrt{\text{Sensitivity} \cdot \text{Specificity}}.
\end{aligned} \tag{5.10}$$

Accuracy is a common metric for assessing classification performance. Sensitivity and specificity measure the classifier's ability to correctly classify "distracted" and "non-distracted" classes, respectively. The performance of both classes are important and the class sizes are not balanced in this study. Therefore, *G*-mean is used to measure the balanced performance between the two classes.

5.4.3 Experimental Settings

One model was constructed and evaluated for each subject, which is called *subject-dependent* models. The detection models were constructed using various machine learning algorithms including SBN-SC, SVM, ELM, LapSVM, SS-ELM, TSVM. SBN-SC [56] is one of the state-of-the-art algorithm used for detecting driver cognitive distraction. SVM has also outperformed the other four algorithms in driver distraction detection study [59]. The supervised learning algorithms, i.e., SBN-SC, SVM, and ELM, constructed the detection model using only labeled sets, while the semi-supervised classification algorithms used both labeled and unlabeled sets. Once the models were constructed, they were all evaluated on the test sets. Details of the model construction by various learning algorithms are described as follows:

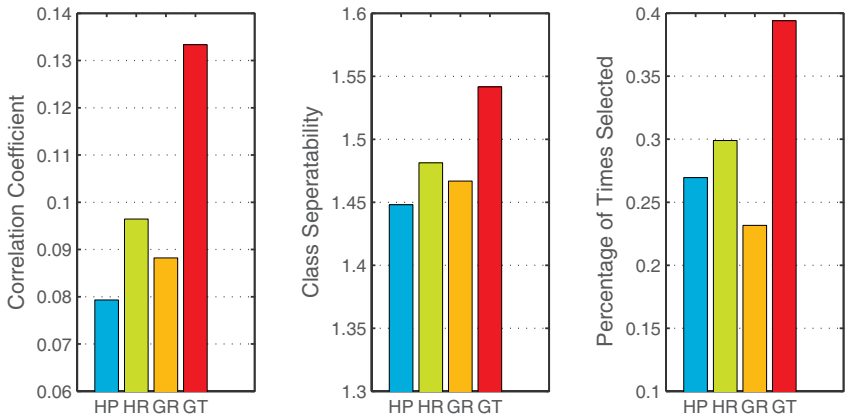
Implementation details The implementations in Matlab toolbox [82] and structure learning package [83] of SBN were used in this study. The supervised clustering method

was implemented in Matlab according to the paper [84]. On this dataset, each feature group in Table 5.1 was first classified into 2-6 classes by supervised clustering. The resulting class indicator of each feature group was the input to one node in SBN. The fast implementation [85, 86] of TSVM was used in this study. The "LibSVM" [87] implementation of SVM was used. The implementation [76] by Melacci and Belkin of LapSVM was used. ELM and SS-ELM were implemented in this study according to [14] and [15], respectively.

Parameter settings The graphs in LapSVM and SS-ELM were constructed using 10 nearest-neighbor method based on Euclidean distance with heat kernel (width equal to 1) weights. The first-degree non-normalized graph Laplacian was used directly. The trade-off coefficient of supervised clustering was selected from $[0.1, 0.2, \ldots, 0.9]$. The topological order of feature of SBN was selected from a candidate set including the orders in Table 5.1 and Figure 5.5(c), and their reverse. The maximum number of parent node of SBN was selected from $[0, 1, 2, 3]$. The number of hidden neurons in ELM and SS-ELM was set to 1000. ELM and SS-ELM were repeated for five times and the average performance was recorded. The Gaussian kernel was used for SVM, LapSVM, and TSVM. The width of Gaussian kernel of SVM and TSVM and the regularization parameters in SVM, TSVM, and SS-ELM were selected from $[10^{-6}, 10^{-5}, \ldots, 10^{6}]$ based on the on validation accuracy,which was obtained via 3 runs of 4-fold cross-validation. Validation data were selected only from the labeled set. The third parameter of LapSVM, i.e., the width of Gaussian kernel, was selected from $[10^{0}, 10^{1}, \ldots, 10^{3}]$ for a balance between computational time and performance. The three-time four-fold validation accuracy was used as the parameter selection criterion.

5.4.4 Feature Analysis

In this experiment, the correlation coefficient and the class separability of each feature were calculated first in order to investigate the importance of the individual feature.



(a) Correlation coefficient (b) Class separability (c) Feedforward feature selection

Figure 5.5: Feature selection results. (HP stands for Head Position, HR stands for Head Rotation, GR stands for Gaze Rotation, and GT stands for Gaze Temporal.)

Figure 5.5(a) and Figure 5.5(b) show the values of each feature group averaged over all subjects and over all features within that feature group. The correlation coefficients of all feature groups were below 0.15, which is very low. Similarity, the class separability was higher than 1 indicating a larger within-class scatter than the between-class scatter. This observation shows that it is difficult to linearly separate the two classes based on a single feature. Comparing different feature groups gives consistent results according to both criteria: Gaze Temporal features are the most important, followed by Head Rotation, Gaze Rotation, and Head Position.

The importance of features when jointly used were further studied via feedforward feature selection, where LDA was used as the basic classifier. For each subject, feedforward feature selection generated a selected feature subset. The importance of a feature was then computed as the percentage of time selected and is shown in Figure 5.5(c). The results show that Gaze Temporal features are again considered as the most important feature group. This is reasonable because all features in this group are advanced features extracted to capture some important patterns of eye behaviors. It is worth noting that all features were selected by at least one subject, indicating that every feature could contribute to the task. Therefore, all features were used in the following experiments.

5.4.5 Comparison with the Related Methods

Table 5.2: Performance comparison of semi-supervised classification methods and supervised learning methods

Measures	SBN-SC	SVM	TSVM	LapSVM	ELM	SS-ELM
Accuracy (%)	83.8 ± 8.5	95.0 ± 4.2	95.5 ± 4.0	97.3 ± 2.9	95.6 ± 3.9	97.2 ± 3.0
Sensitivity (%)	89.2 ± 9.1	97.3 ± 3.6	97.3 ± 3.1	98.8 ± 2.8	97.1 ± 3.7	98.2 ± 3.2
Specificity (%)	69.6 ± 20.2	88.6 ± 12.0	90.5 ± 8.0	92.8 ± 9.4	91.2 ± 8.9	93.9 ± 7.9
G-mean	0.773 ± 0.144	0.926 ± 0.073	0.938 ± 0.053	0.956 ± 0.054	0.940 ± 0.053	0.959 ± 0.045
Training Time (ms)	291.1	0.5	273.0	227.9	29.7	150.4
Prediction Time (ms)	35.2	0.2	8.7	1.8	4.4	4.5

In this experiment, the size of the labeled set was set to 60, which is equivalent to labeling up to 10-minute driving. The rest of the training set formed the unlabeled set. Table 5.2 shows the performance (mean $\pm$ standard deviation) of all the algorithms. The first observation is that SBN-SC achieved much lower performance compared with the other algorithms. One possible reason is that, as a generative model, SBN-SC tries to model the feature distribution given a class in the form of conditional probability functions. However, in this case, many conditioning cases were represented by too few training samples, which could not provide sufficient basis for estimating the conditional probability functions. In contrast, all the other discriminative models achieved G-mean of more than 0.9, demonstrating their effectiveness in constructing driver distraction detection systems. A further comparison among the five discriminative models showed that TSVM and LapSVM outperformed SVM, and similarly SS-ELM outperformed ELM with statistical significance (Right-tailed paired t-test on G-mean, TSVM v.s. SVM: $t(33) = 2.9, p < 0.01$; LapSVM v.s. SVM: $t(33) = 4.5, p < 0.01$, SS-ELM v.s. ELM: $t(33) = 4.0, p < 0.01$). Moreover, TSVM showed less improvement than LapSVM.

Furthermore, the computational time was recorded and compared. A PC with Intel Core$^{\text{TM}}$ i5-2520M 2.50 GHz processor and 12 GB RAM was used to run all the algorithms. Table 5.2 presents the training and prediction time in milliseconds. All algorithms are suitable for real-time detection, as they took only up to tens milliseconds. In

terms of training time, SBN-SC took the longest time because of the supervised clustering process. The semi-supervised classification algorithms took longer training time than their supervised origins.

5.4.6 Influence of Unlabeled Data

Motivated by the promising performance of the graph-based method in the previous section, the focuses of the experiments were on the graph-based methods from this section onwards. This experiment test the graph-based semi-supervised classification algorithms' performance under conditions with different sizes of the unlabeled set. The size of labeled set was fixed to 60, and the size of unlabeled set was set to $[0, 120, 240, 360, 480]$ in several trails.

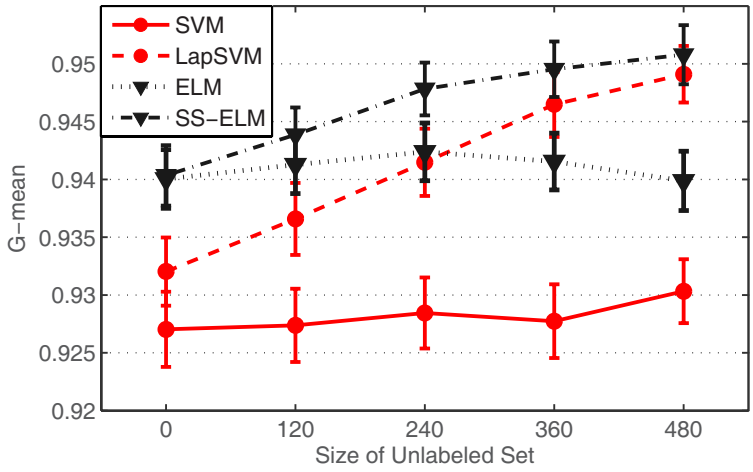


Figure 5.6: G-mean averaged across all subjects using sizes of unlabeled set. (The error bars indicate standard error of mean.)

Figure 5.6 shows the plot of the performance against different conditions. The error bar indicates the standard error of the mean, which is calculated as the standard deviation of the sample divided by the square root of the size of the sample. The standard deviation of the sample measures how spread out numbers are from the average, while the standard error of the mean measures how precise the sample mean is in estimating the population mean. Since here the main focus is to compare the mean obtained under

different conditions, this plot uses the standard error of the mean instead of the standard deviation of the sample as in Table 5.2.

The performance of LapSVM and SS-ELM under the five conditions with increasing size of unlabled set is shown in Figure 5.6. SVM and ELM make use of only the labeled data so their performance did not change significantly with the size of unlabeled set. In contrast, LapSVM and SS-ELM had significant increase in performance when more unlabeled data points were used for training. A further comparison between the semi-supervised classification algorithms and their origins showed that LapSVM outperformed SVM statistically in all cases. SS-ELM outperformed ELM when the size of unlabeled set was larger or equal to 240, while there was no statistical difference between the performance of ELM and SS-ELM in cases where the size of unlabeled set was less than 240.

5.4.7 Influence of Labeled Data

In this experiment, the size of unlabeled set was set to 45, while the size of labeled set increased from 16 to 80 as shown in the horizontal axis of Figure 5.7. The G-mean is plotted against the size of labeled set. All the four algorithms showed increasing detection performance as the size of labeled set increased. Furthermore, LapSVM and SS-ELM outperformed SVM and ELM, respectively, in all cases. If a certain value of G-mean is required, LapSVM and SS-ELM require a smaller size of labeled set compared with SVM and ELM, respectively.

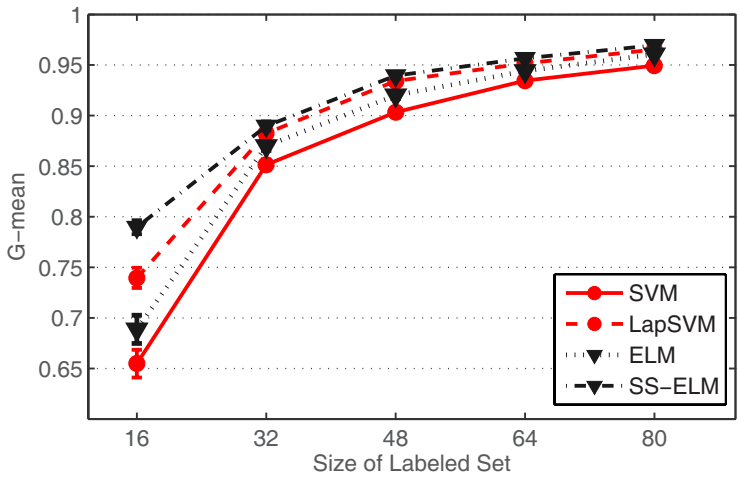


Figure 5.7: G-mean averaged across all subjects using different sizes of labeled set. (The error bars indicate standard error of mean.)

5.4.8 Influence of Hyper-Parameters

SS-ELM has several advantages over LapSVM, such as able to handle multi-class classification problem naturally and having fewer hyper-parameters to tune. In this experiment, SS-ELM was further studied in terms of its sensitivity to hyper-parameters. The same setting of labeled and unlabeled set was used as in Section 5.4.5 and one subject was used as an example. The performance under different settings of hyper-parameter settings was obtained and recorded.

Firstly, the model related parameters, i.e., the penalty coefficient for manifold regularization term λ and the penalty coefficient for empirical training error C, were investigated. Figure 5.8 shows the G-mean of SS-ELM under different settings. The performance ranged from 0 to 1 suggesting that the performance of SS-ELM is sensitive to the model related parameters. It can also be observed that G-mean remained over 0.9 over a large and continuous region. The performance was good as long as C was set to a large value and even better performance was achieved when λ was set to a large value. Therefore, cross validation should be sufficient to choose the value for model related parameters.

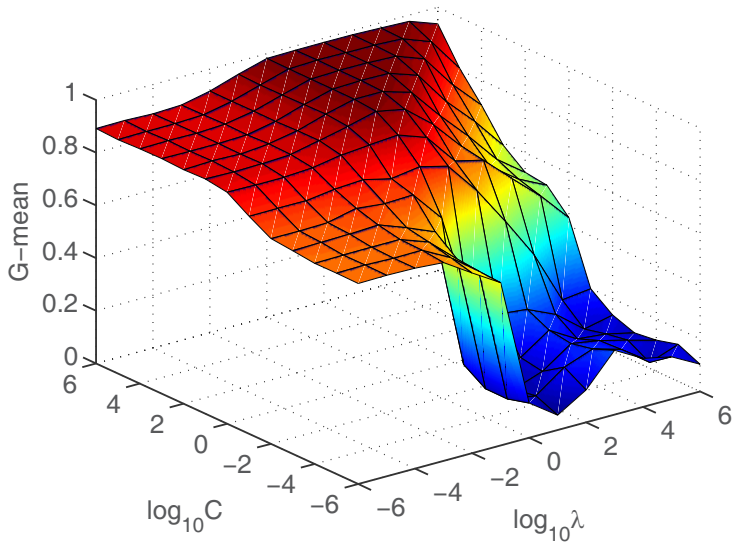


Figure 5.8: G-mean of SS-ELM using 60 labeled data and maximum number of unlabeled data under different regularization parameters. (Warm colors indicate higher values of G-mean, while color colors indicate lower values of G-mean.)

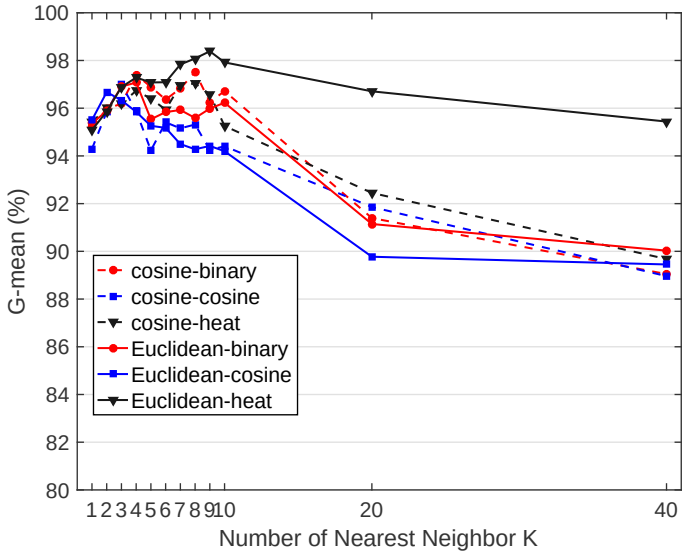


Figure 5.9: G-mean of SS-ELM using 60 labeled data and maximum number of unlabeled data under different graph-related parameters. (The legend indicates " distance metrics - weighting schemes.")

Secondly, the graph related parameters, i.e., the number of nearest neighbors K, the distance metrics and the weighting schemes, were investigated. Figure 5.9 presents the G-mean of SS-ELM under different graph-related parameter settings. The combination of using Euclidean distance metric and heat kernel as the weighting scheme generated better performance than the other settings. Moreover, the performance fluctuated within a small range as long as the number of nearest neighbor K was set to a value between 4 and 20. Another observation is that all lines show decreasing trends when K is larger than 10. The decreasing performance under large value of K and the superior performance of heat kernel are also observed in another study [88] on the general datasets.

5.4.9 Discussion

Section 5.4.4 analyzed the importance of features individually and jointly and found the gaze temporal features are the most contributing features to driver distraction detection. Therefore, it is suggested to include and exploit such features in future studies on cognitive distraction detection.

Different from most existing research, which collected data in a driving simulator environment, this thesis used data from an on-road experiment. Drivers' behaviors in on-road driving scenarios can be very different from those in driving simulator scenarios because of the boarder view and the more complex environment. The satisfactory detection performance of LapSVM and SS-ELM, as shown in Section 5.4.5, suggests that the relationship between drivers' eye and head movements and the distraction state is still very prominent in the on-road driving scenarios. Furthermore, compared with the supervised methods, the semi-supervised classification methods can improve the performance of distraction detection by exploiting the data structure without actually labeling the data.

Section 5.4.5 also compares the graph-based methods and the low-density-separation-based methods on the driver distraction detection task. These two types of methods make

different assumptions on the underlying data structure, i.e., manifold-like and cluster-like, both of which can be reasonable depending on the problem. The superior performance of the graph-based methods over the low-density-separation-based methods implies that the eye and head movement data form some manifold structures. Therefore, the graph-based algorithms are recommended for developing driver distraction detection system. Other graph-based methods, such as those in unsupervised learning setting, are also recommended for studying driver distraction using eye and head movements. Section 5.4.6 also tests the computational cost of each algorithm and confirms the real-time capability of the detection system in a PC-based Matlab setting.

As shown in Section 5.4.6 and Section 5.4.7, the performance of the graph-based algorithms was affected by the size of the labeled set and the unlabeled set jointly. The difference between the semi-supervised classification algorithms and the supervised learning algorithm increased with the size of the unlabeled set. In the context of driver distraction detection using eye and head movement, the unlabeled data can be collected with a relatively low cost from naturalistic driving. The performance of semi-supervised classification algorithms can be further improved by adding more unlabeled training data at a low cost. On the other hand, the difference decreased with the size of labeled set as shown in Figure 5.7. Therefore, the graph-based algorithms are recommended especially for cases where a large amount of unlabeled data and limited labeled data are available.

The results in Section 5.4.7 also suggest that the graph-based algorithms require much less labeled data in order to achieve the same performance as the supervised learning algorithms. It is not only costly but also erroneous to label training data due to the yet incomplete domain knowledge on the human distraction. Therefore, by reducing the amount of labeled data, the graph-based semi-supervised classification methods can reduce the labeling cost. In addition, with smaller amount of labeled data required, a high-quality data collection is made more feasible.

In Section 5.4.8, satisfactory results of SS-ELM were observed from multiple regular-

ization parameter values, suggesting that the performance is robust as long as the parameters are within that range. Cross validation should be sufficient to select the value for regularization parameters. Moreover, the performance varied only in a small range as long as K is between 4 and 20. Therefore, the graph-related parameters can be fixed to 10-nearest-neighbor graph based on Euclidean distance with weights computed using heat kernel.

5.5 Experiments of Clustering-based Labeling Assistance

The objective of the experiment is to investigate the performance of the two proposed clustering algorithms, ELM-JEC and ELM-CLR, in comparison with their competitive algorithms. In the following sections, the evaluation metrics are first defined, followed by the investigations of ELM-JEC and ELM-CLR, separately.

5.5.1 Evaluation Metrics

Since in driver distraction dataset, the two classes, distracted driving and normal driving, are slightly imbalanced, the evaluation metric should take into account the detection performance for each class. The clustering G-mean is used and defined as:

$$\begin{aligned} \text{Sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}}, \\ G\text{-mean} &= \sqrt{\text{Sensitivity} \cdot \text{Specificity}}. \end{aligned} \tag{5.11}$$

Moreover, the true positive (TP), true negative (TN), false positive (FP), and false nega-

tive (FN) in the clustering setting are first defined as follows:

$$\begin{aligned}
\text{TP} &= \textstyle\sum_1^n \delta\left(t_i, \text{map}(t_i^p)\right) \cdot \delta\left(\text{map}(t_i^p), 1\right), \\
\text{TN} &= \textstyle\sum_1^n \delta\left(t_i, \text{map}(t_i^p)\right) \cdot \delta\left(\text{map}(t_i^p), 0\right), \\
\text{FP} &= \textstyle\sum_1^n \tilde{\delta}\left(t_i, \text{map}(t_i^p)\right) \cdot \delta\left(\text{map}\left(t_i^p\right), 1\right), \\
\text{FN} &= \textstyle\sum_1^n \tilde{\delta}\left(t_i, \text{map}(t_i^p)\right) \cdot \delta\left(\text{map}(t_i^p), 0\right),
\end{aligned} \tag{5.12}$$

where n is the size of the dataset, $t_i \in \{0, 1\}$ is the ground truth class label, a value of one represents the distracted driving class and a value of zero represents the normal driving class, map(t_i^p) is a mapping function that maps each cluster label in the clustering solution to a ground truth class label such that the resulting G-mean achieves its maximum value. $\delta(a, b) = 1$ if $a = b$; $\delta(a, b) = 0$, otherwise. $\tilde{\delta}(a, b) = 1$ if $a \neq b$; $\tilde{\delta}(a, b) = 0$, otherwise.

Here in unsupervised setting, the main goal is to distinguish different groups of objects but not necessarily to map the group to any exact class label. Therefore, note that Equation (5.12) wraps a mapping function around the clustering solution, instead of directly using the predicted label as in the definition of G-mean in supervised learning setting in Section 5.4.2. This mapping function maps the groups to class labels such that the larger value of G-mean is achieved and used as the final evaluation metric.

5.5.2 Experimental Settings

The proposed algorithm ELM-JEC was compared with several embedding-based clustering algorithms, k-means [61], LDA-KM [31], ELMC-LDA [35], DEC [11], and US-ELM [15]. Details of the parameter setting are as follows:

• In the ELM-based methods, 1) the number of hidden neurons was set to 500, 2) the sigmoid function was used as the activation function.

• The trade-off parameters in LDA-KM, ELMC-LDA, US-ELM, DEC, and ELM-JEC were selected from $[2^{-4}, \ldots, 2^4]$.

• The graph Laplacian used by US-ELM and ELM-JEC was constructed using the 5-nearest -neibhor graph with binary weight.

• The dimensionalities of embedded space used in US-ELM, DEC, and ELM-JEC was set to one.

• The iterative methods, i.e., LDA-KM and ELM-JEC, were initialized with outputs of k-means and were terminated when there was no change in the predicted labels, or a maximum number of iteration (20 in the experiment) was reached.

• The algorithms with inherent randomness were run for 50 times independently.

• The data were standardize to have zero mean and unit variance.

• Leave-one-subject-out cross validation was used to select the optimal values of hyper-parameters.

The proposed algorithm ELM-CLR was compared with several graph-based clustering methods, SC [71], NCut [12], CAN [16], and PCAN [16]. In addition, another ELM-based clustering method, i.e., US-ELM [15], was also implemented and compared in the study. Details of the parameter setting are as follows:

• The algorithms with unstable results, i.e., k-means, SC, NCut, US-ELM, and ELM-CLR, were run for 50 times independently.

• The number of eigenvectors used in SC and NCut and the dimensionality of the embedded space in PCAN, US-ELM, and ELM-CLR was set to one.

• In the algorithms based on ELM, i.e., US-ELM and ELM-CLR, 1) the sigmoid function was used as the activation function, 2) the number of hidden neurons was set to 500, 3) the regularization parameter was selected from $[0.0001, 0.001, \ldots, 10000]$.

• In the fixed graph based clustering algorithms, i.e., SC, NCut, and US-ELMa (to distinguish from the US-ELM with manually constructed graph), the graph was constructed automatically using a self-tuning Gaussian method.

• In CAN and PCAN, the graph updating schemes (i.e., updating all weights or only the weights with nearest neighbors) were selected. In ELM-CLR, all weights were updated.

• In all graph based algorithms, the number of neighbors was selected from [1 6, 11, 16, 21, 26, 31].

• The data was standardize to have zero mean and unit variance.

• The leave-one-subject-out cross validation was used to select the values of hyper-parameters.

5.5.3 Results of ELM-JEC

Table 5.3 shows the Clustering G-mean of ELM-JEC and the related algorithms. With significance level of 5%, the proposed algorithm ELM-JEC outperformed all the other algorithms. This observation confirms our hypothesis that class separability and local structure preserving are both important for joint embedding and clustering on driver distraction dataset. In the context of driver distraction study, ELM-JEC achieved G-mean of around 0.70. The only human intervention needed to obtain such result is the leave-one-subject-out cross validation, where no label information on the current subject is assumed but the full access to the label set for all other subjects is required. This kind of human intervention can be achieved by letting human experts first label a few subject's driving data to form the labeled validation set.

Table 5.3: G-mean of ELM-JEC and the related algorithms on driver distraction dataset

	Clustering G-mean	# of subject where ELM-JEC is statistically better	# of subject where ELM-JEC is statistically worse	# of subject where there is no statistical difference with ELM-JEC	p-value of one-tailed paired t-test with ELM-JEC
k-means	0.5732 ± 0.1372	28	2	4	$<0.0001*$
LDA-KM	0.5812 ± 0.1462	26	3	4	$<0.0001*$
ELMC-LDA	0.6686 ± 0.1081	17	9	4	$0.0068*$
DEC	0.6260 ± 0.1628	18	5	4	$<0.0001*$
US-ELM	0.6581 ± 0.1310	18	14	2	$0.0480*$
ELM-JEC	0.6994 ± 0.1286	-	-	-	-

* denotes that ELM-JEC is statistically better with the significance level of 5%.

5.5.4 Results of ELM-CLR

Table 5.4 shows the G-mean of ELM-CLR and the related algorithms on the driver distraction dataset. There were 34 subjects in total. Comparing ELM-CLR with the other algorithms using one-tailed paired t-test shows that ELM-CLR was statistically better than k-means, SC, NCut, CAN, and PCAN at a significance level of 5%. This demonstrates that using dual representations for adaptive graph learning is more effective for clustering driver distraction. However, comparing ELM-CLR with US-ELM^a shows that there was no statistically significant difference overall. Nevertheless, out of 34 subjects, ELM-CLR outperformed US-ELM^a on 16 subjects and underperformed on 12 subjects.

Table 5.4: G-mean of ELM-CLR and the related algorithms on driver distraction dataset

	G-mean	# of subject where ELM-CLR is statistically better	# of subject where ELM-CLR is statistically worse	# of subject where there is no statistical difference with ELM-CLR	p-value of one-tailed paired t-test with ELM-CLR
k-means	0.5732 ± 0.1372	28	3	3	<0.0001 *
SC	0.3236 ± 0.1777	30	4	0	<0.0001 *
NCut	0.6195 ± 0.1561	22	12	0	0.0267 *
CAN	0.2872 ± 0.1989	31	0	3	<0.0001 *
PCAN	0.5086 ± 0.1937	23	4	7	<0.0001 *
US-ELM^a	0.6782 ± 0.1207	16	12	6	0.4476 $^-$
ELM-CLR	0.6801 ± 0.1535	-	-	-	-

* denotes that ELM-CLR is statistically better with the significance level of 5%.

$^-$ denotes that there is no statistical difference with ELM-CLR.

5.6 Summary

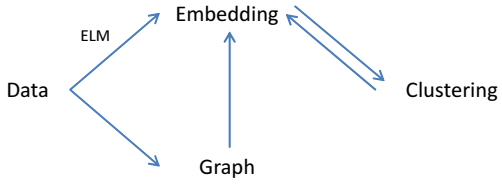
This chapter addresses the second objective of the thesis, i.e., to propose a semi-supervised driver distraction detection system with efficient labeling assistance. The need for empirically examining the two types of semi-supervised classification algorithms is established in Section 5.1. It is hypothesized that the graph-based algorithms, which are reviewed in Section 5.2, are more suitable for driver distraction detection. The perfor-

mance of all the related supervised, semi-supervised, and unsupervised learning algorithms is evaluated and compared based on the data set described in Section 5.3. Highlights of the findings from the results in Section 5.4 include 1) the eye temporal features were the most dominating features for detecting driver distraction; 2) the graph-based algorithms outperformed the low-density-separation-based algorithms; 3) the graph-based algorithm showed the most improvement when the labeled data were limited and the unlabeled data were plentiful. 4) In terms of the clustering performance, ELM-JEC outperformed all the related methods with statistical significance. 5) ELM-CLR performed on a par with US-ELM with automatically constructed graph but outperformed the other related clustering algorithms.

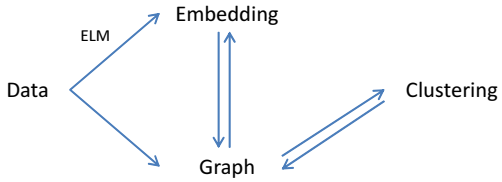
Chapter 6

Conclusions and Future Works

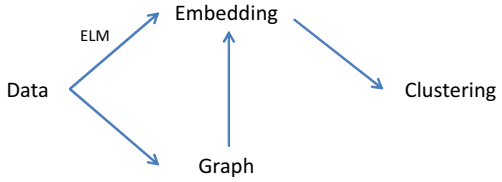
Based on the work presented in Chapter 3, Chapter 4, and Chapter 5, Chapter 6 draws conclusions regarding the two objectives defined in Chapter 1 and recommends future research directions.



(a) ELM-JEC



(b) ELM-CLR



(c) US-ELM

Figure 6.1: The relationship between data, embedding, and graph

This chapter concludes the thesis from five aspects as listed in the following subsections. Each subsection begins with a statement of the main contribution in italics. Also, the details of the conclusion and further work are described in each subsection.

(1) ELM can benefit the task of clustering by learning a nonlinear embedding of the data. Since the hidden neurons of ELM are randomly generated and fixed, the learning of embedding is equivalent to solving only the output weights such that the embedding has some desirable properties.

To improve the clustering performance by taking advantage of ELM is the first objective. The thesis meets this objective by using ELM as an effective nonlinear embedding tool. Under this framework, the thesis develops two clustering algorithms, i.e., ELM-JEC and ELM-CLR. The difference between them is shown in Fig. 6.1(a) and Fig. 6.1(b). The embedding in ELM-JEC is affected by data, graph, and the clustering results, while the embedding in ELM-CLR is affected by the data and the graph. In terms of how ELM-

based embedding benefit the task of clustering, the embedding in ELM-JEC directly affects the clustering result, while the embedding in ELM-CLR affects the clustering result via graph. The relationship in US-ELM is also illustrated for comparison purpose. As shown in Fig. 6.1(c), US-ELM embedding is only a result of the data and the graph.

Different from traditional ELM-based methods, where ELM alone is exploited to handle a complete machine learning task, such as classification, the thesis proposes to use ELM to solve an intermediate machine learning problem, i.e., to obtain an embedding with the ultimate goal of clustering. To investigate the possibility of using other methods jointly with ELM to solve complex machine learning problems is the future work.

(2) Both structure preserving and separability maximizing are necessary properties of embedding under the goal of clustering.

As reported in Chapter 3, ELM-JEC produced comparable or better clustering results than both US-ELM and DEC. US-ELM is an example of embedding-based clustering method that preserves the structure of the data. DEC is an example that maximizes the separability. Therefore, the thesis concludes that both properties are necessary under the goal of clustering.

In the future work, other ways of formulating the desirable properties in Table 3.1 should be studied. In order to apply the proposed algorithms to real-world applications, future works also include investigating effective hyper-parameter selection methods and strategies that prevent the iterative optimization algorithms from going into a local minimum.

(3) Combining the nonlinear structure-preserving embedding with the original data as the basis of graph learning is beneficial for clustering.

As reported in Chapter 4, ELM-CLR outperformed these three algorithms on most of the datasets. CAN and CLR use original data for graph learning, and PCAN uses linearly projected data for graph learning. Therefore, the thesis concludes that combining the nonlinear structure-preserving embedding with the original data is beneficial for clustering.

The graph is widely used in many machine learning tasks, such as feature selection and classification. It is worth investigating whether the use of nonlinear structure-preserving embedding for graph learning is also beneficial for the other machine learning tasks.

(4) Towards the goal of making use of unlabeled training data, the thesis concludes that graph-based semi-supervised classification methods are more suitable for eye-movement-based driver distraction detection than low-density-separation-based methods.

As presented in Chapter 5, the graph-based semi-supervised classification algorithms achieve a better detection G-mean of 0.0245 compared with the traditional supervised learning counterparts. Therefore, the thesis concludes that graph-based methods are more suitable for driver distraction detection. Moreover, taking all aspects of the performance into consideration, this thesis proposes to use SS-ELM for the semi-supervised detection system, which 1) achieves detection performance of 0.959 G-mean, 2) has potential to achieve better performance if more unlabeled training data are available, 3) shows increasingly advantages over supervised learning based counterpart with fewer labeled training data.

One limitation is that the proposed system was only tested on the data collected in the same experiment, i.e., the same driver doing the car-following task on trunk roads. The detection is expected to be less accurate if the system is directly used in a different environment, e.g., for a different driver or in a more complex driving scenario. Therefore, it is worth investigating driver-independent distraction detection system that adapts to new drivers. Towards this goal, at least three works should be explored: 1) More training data from different scenarios are needed in order to develop and test a more general detection system; 2) *Online learning* algorithms [89] update the model gradually as more training data become available and should be added to the distraction detection system in order for the system to continuously make use of incoming data; 3) *Transfer learning* algorithms [90] handle the problem, where training and test data are different but related, and should also be added in order for the system to handle unseen drivers and unseen scenarios.

(5) Towards the goal of labeling assistance, the thesis recommends ELM-JEC for its better clustering accuracy if reliable hyper-parameter selection can be achieved, and ELM-CLR, otherwise.

In ELM-JEC, the development of embedding takes into account the performance of the clustering. In the task of clustering the eye and head movements for driver distraction labeling, ELM-JEC achieved better clustering performance than ELM-CLR. Meanwhile, the advantage of ELM-CLR is that the there is no need to construct manually the similarity graph between data points a prior. As a result, the users need to select fewer hyper-parameters. Taking this trade-off between clustering accuracy and hyper-parameter selection, the thesis recommends 1) ELM-JEC for its better clustering accuracy, if reliable parameter selection can be achieved; 2) ELM-CLR, otherwise.

In the driver distraction detection, the desired number of clusters is two, because clustering serves the purpose of labeling assistance and the downstream classification task has two classes, i.e., distracted driving and normal driving. In some other applications, the constraint on the number of clusters is milder leaving little information for human to manually determine the number of clusters. Methods that can automatically determine the number of clusters are needed. For these applications, we find the following promising approaches for modifying the proposed algorithm:

Approach 1: Wrap a cluster-number-selection method around the proposed algorithm

ELM-JEC or ELM-CLR can be repeated for different numbers of clusters. Some fitness criterion can be calculated with clustering results under each number of clusters. The final cluster number can be selected as the number with the best fitness criterion value.

Approach 2: Select cluster number during ELM-CLR clustering

In the previous approach, the proposed algorithm is independently run with different number of clusters, which can be time consuming. The proposed algorithm, ELM-CLR starts with a full connected graph (cluster number = 1) and iteratively makes the graph less connected (increasing the number of clusters) until the specified number of

clusters is achieved. If the number of cluster is k, in each iteration of ELM-CLR, a set of clustering result is generated with number of cluster $\leq k$. Therefore, the evaluation of number of clusters can be done during ELM-CLR clustering process by introducing a fitness value for each number of cluster.

In both two approaches, the research question that needs to be answer is how to compute the fitness value Q of each cluster number. One common example of such fitness value is Separation Index (SI). A gap statistic has been proposed [91] and is suitable for almost all clustering algorithms. In X-means, the fitness value is computed by Bayesian Information Criterion under the assumption that the cluster models are all spherical Gaussians [92]. For clustering algorithms based on models, some other information criteria [93] can be applied, such as Akaike's Information Criterion (AIC), Approximate Weight of Evidence (AWE), Classification Likelihood Criterion (CLC), and Kullback Information Criterion (KIC). In robust information clustering, the criterion is proposed based on real risk VC-bound [94]. The advantage of ELM-CLR is that the criterion can be calculated based not only on the data in the original space, but also the data in the embedded space and the graph learned. We will investigate these research directions in the future work.

Another case that should be considered in the future work is that the desired number of clusters (i.e., the number of classes required by the downstream tasks) is smaller than the number of true clusters formed by the data. We have shown in Section 4.3.8 that ELM-CLR will merge two clusters (whose samples are similar) as one. If more detailed information about the correspondence between class and clusters is needed, one possible approach, named two-layer clustering strategy, is explained as follows:

In the first layer, the proposed algorithm is used to cluster data to a suitable number of clusters. Then a similarity matrix of the clusters can be built by considering the pairwise similarity between data from one cluster and data from another cluster, either in the original space or the embedded space. In the second layer, each cluster can be treated as a data point, and similarity-based clustering algorithm like spectral clustering can be

applied to cluster these data points into a suitable number of clusters.

This approach is proposed in a heuristic manner. Future experimental studies are needed to verify the effectiveness of this approach.

Author's Publications

Journal Papers:

1. Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. "Extreme Learning Machine for Joint Embedding and Clustering," *Neurocomputing*, vol. 277, pp. 78-88, 2018.

2. Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. "An adaptive graph learning method based on dual data representations for clustering," *Pattern Recognition*, vol. 77, pp. 126-139, 2018.

3. Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. "Driver Distraction Detection Using Semi-Supervised Machine Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1108-1120, 2016.

4. Gao Huang, Tianchi Liu, Yan Yang, Shiji Song, and Cheng Wu. "Discriminative clustering via extreme learning machine", *Neural Networks*, vol. 70, pp. 1-8, 2015.

5. Tianchi Liu, Zhiping Lin, Marcus Eng Hock Ong, Zhi Xiong Koh, Pin Pin Pek, Yong Kiang Yeo, Beom-Seok Oh, Andrew Fu Wah Ho, and Nan Liu. "Manifold ranking based scoring system with its application to cardiac arrest prediction: A retrospective study in emergency department patients", *Computers in Biology and Medicine*, vol. 67, pp. 74-82, 2015.

Conference Papers:

1. Tianchi Liu, Yue Li, Zuo Bai, Jaydeep De, Cao Vinh Le, Zhiping Lin, Shih-Hsiang Lin, Guang-Bin Huang, and Dongshun Cui. "Two-stage structured learning approach for stable occupancy detection," In *Proceedings of IEEE International Joint Conference on Neural Networks,* 2016, pp. 2306-2312.

2. Tianchi Liu, Yan Yang, Guang-Bin Huang, and Zhiping Lin. "Cluster Regularized Extreme Learning Machine for Detecting Mixed-Type Distraction in Driving," In *Proceedings of IEEE International Conference on Intelligent Transportation Systems,* 2015, pp. 1323 - 1326.

3. Chamara Kasun Liyanaarachchi Lekamalage, Tianchi Liu, Yan Yang, Zhiping Lin, and Guang-Bin Huang. "Extreme Learning Machine for Clustering," In *Proceedings of ELM-2014 Volume 2,* 2015, pp. 379-3872.

4. Yan Yang, Haoqi Sun, Tianchi Liu, Guang-Bin Huang, and Olga Sourina. "Driver workload detection in on-road driving environment using machine learning," In *Proceedings of ELM-2014 Volume 2,* 2015, pp. 379-387.

Bibliography

[1] M. Manktelow, "History of taxonomy," *Lecture from Dept. of Systematic Biology, Uppsala University*, 2010.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[3] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[4] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster analysis*, 5th ed. John Wiley & Sons, 2011.

[5] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*, 2nd ed. London, UK: MIT Press, 2006.

[6] V. Garla, C. Taylor, and C. Brandt, "Semi-supervised clinical text classification with laplacian svms: An application to cancer case management," *Journal of Biomedical Informatics*, vol. 46, no. 5, pp. 869 – 875, 2013.

[7] L. Gómez-Chova, G. Camps-Vails, J. Muñoz-Marí, and J. Calpe, "Semi-supervised cloud screening with Laplacian SVM," in *Proc. IEEE Int. Conf. Geoscience and Remote Sensing Symp.* IEEE, 2007, pp. 1521–1524.

[8] Y. Liang, M. L. Reyes, and J. D. Lee, "Real-Time Detection of Driver Cognitive Distraction Using Support Vector Machines," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 340–350, Jun. 2007.

[9] N. Li and C. Busso, "Predicting perceived visual and cognitive distractions of drivers with multimodal features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 51–65, 2015.

[10] Y. Zhang, Y. Owechko, and J. Zhang, "Driver cognitive workload estimation: A data-driven perspective," in *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2004, pp. 642–647.

[11] C. Hou, F. Nie, D. Yi, and D. Tao, "Discriminative Embedded Clustering: A Framework for Grouping High-Dimensional Data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1287–1299, June 2015. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2014.2337335

[12] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[13] G.-B. Huang, L. Chen, C. K. Siew *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, 2006.

[14] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[15] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-Supervised and Unsupervised Extreme Learning Machines," *IEEE Tran. on Cybern.*, vol. 44, no. 12, pp. 2405–2417, Dec 2014.

[16] F. Nie, X. Wang, and H. Huang, "Clustering and Projected Clustering with Adaptive Neighbors," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014, pp. 977–986.

[17] F. Nie, X. Wang, M. I. Jordan, and H. Huang, "The constrained laplacian rank algorithm for graph-based clustering," in *Proc. 30th AAAI*, 2016.

[18] T. Liu, Y. Yang, G. B. Huang, Y. K. Yeo, and Z. Lin, "Driver Distraction Detection Using Semi-Supervised Machine Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1108–1120, April 2016.

[19] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, vol. 14, 2002, pp. 585–591.

[20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489 – 501, 2006.

[21] L. Zhang and D. Zhang, "Domain Adaptation Extreme Learning Machines for Drift Compensation in E-Nose Systems," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 7, pp. 1790–1801, July 2015.

[22] J. Cao, W. Wang, J. Wang, and R. Wang, "Excavation Equipment Recognition Based on Novel Acoustic Statistical Features," *IEEE Trans. Cybern.*, vol. PP, no. 99, pp. 1–13, 2016.

[23] B. Wang, S. Wang, X. Liu, and J. Yang, "Effective object tracking using extreme learning machine with smoothness and preference regularisation," *Electron. Lett.*, vol. 51, no. 23, pp. 1867–1869, 2015.

[24] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, 4th ed. Wiley Publishing, 2009.

[25] L. J. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," *J. Mach. Learn. Res.*, vol. 10, no. 1-41, pp. 66–71, 2009.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: http://dx.doi/org/10.1126/science.290.5500.2323

[28] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[29] C. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and k-means clustering," in *Proc. 24th ICML*, 2007, pp. 521–528.

[30] H. Zhang, Y. Zhuang, and F. Wu, "Cross-modal correlation learning for clustering on image-audio dataset," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07. New York, NY, USA: ACM, 2007, pp. 273–276. [Online]. Available: http://doi.acm.org/10.1145/1291233.1291290

[31] J. Ye, Z. Zhao, and M. Wu, "Discriminative K-means for Clustering," in *Advances in Neural Information Processing Systems*, vol. 7, 2007, pp. 1649–1656.

[32] Q. He, X. Jin, C. Du, F. Zhuang, and Z. Shi, "Clustering in extreme learning machine feature space," *Neurocomputing*, vol. 128, pp. 88–95, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2012.12.063

[33] Y. Peng, W.-L. Zheng, and B.-L. Lu, "An unsupervised discriminative extreme learning machine and its applications to data clustering," *Neurocomputing*, vol. 174, Part A, pp. 250 – 264, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2014.11.097

[34] C. K. Liyanaarachchi Lekamalage, T. Liu, Y. Yang, Z. Lin, and G.-B. Huang, *Extreme Learning Machine for Clustering*. Cham: Springer International Publishing, 2015, pp. 435–444. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-14063-6_36

[35] G. Huang, T. Liu, Y. Yang, Z. Lin, S. Song, and C. Wu, "Discriminative clustering via extreme learning machine," *Neural Networks*, vol. 70, pp. 1 – 8, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2015.06.002

[36] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006.

[37] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. John Wiley & Sons, 2000.

[38] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recogn.*, vol. 41, no. 1, pp. 176 – 190, 2008.

[39] K. Tasdemir, B. Yalçin, and I. Yildirim, "Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures," *Pattern Recogn.*, vol. 48, no. 4, pp. 1465 – 1477, 2015.

[40] J. Hou, W. Liu, X. E, and H. Cui, "Towards parameter-independent data clustering and image segmentation," *Pattern Recogn.*, vol. 60, pp. 25 – 36, 2016.

[41] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. 17th NIPS*. Cambridge, MA, USA: MIT Press, 2004, pp. 1601–1608.

[42] L. Du and Y.-D. Shen, "Unsupervised feature selection with adaptive structure learning," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2015, pp. 209–218.

[43] X. Wang, Y. Liu, F. Nie, and H. Huang, "Discriminative unsupervised dimensionality reduction," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 3925–3931.

[44] L. Gao, J. Song, F. Nie, F. Zou, N. Sebe, and H. T. Shen, "Graph-without-cut: An ideal graph learning for image segmentation," in *Proc. 30th AAAI*, 2016, pp. 1188–1194.

[45] X. Zhu, "Semi-Supervised Learning Literature Survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Tech. Rep. 1530, 2005.

[46] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," in *Proc. 16th ICML.* San Francisco, CA: Morgan Kaufmann, 1999, pp. 200–209.

[47] O. Chapelle, M. Chi, and A. Zien, "A Continuation Method for Semi-supervised SVMs," in *Proc. 23rd ICML.* New York: ACM, 2006, pp. 185–192.

[48] M. Meytlis and L. Sirovich, "On the dimensionality of face space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 7, 2007.

[49] P. Jiménez, L. Bergasa, J. Nuevo, N. Hernández, and I. Daza, "Gaze Fixation System for the Evaluation of Driver Distractions Induced by IVIS," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1167–1178, Sep. 2012.

[50] A. Ueno, S. Tei, T. Nonomura, and Y. Inoue, "An analysis of saccadic eye movements and facial images for assessing vigilance levels during simulated driving," *Engineering Psychology and Cognitive Ergonomics*, pp. 451–460, 2009.

[51] K. Young, J. D. Lee, and M. A. Regan, *Driver distraction: Theory, effects, and mitigation*. CRC Press, 2008.

[52] O. Nakayama, T. Futami, T. Nakamura, and E. R. Boer, "Development of a steering entropy method for evaluating driver workload," SAE, Detroit, MI, Tech. Rep. 1999-01-0892, Mar. 1999.

[53] M. Wöllmer, C. Blaschke, T. Schindl, B. Schuller, B. Färber, S. Mayer, and B. Trefflich, "Online Driver Distraction Detection Using Long Short-Term Memory," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 574–582, Jun. 2011.

[54] Y. Yang, "The Effects of Increased Workload on Driving Performance and Visual Behaviour," Ph.D. dissertation, University of Southampton, 2011.

[55] Y. Liang, J. D. Lee, and M. L. Reyes, "Nonintrusive detection of driver cognitive distraction in real time using Bayesian networks," *J. of Transp. Res. Rec.*, vol. 2018, no. 1, pp. 1–8, Dec. 2007.

[56] Y. Liang and J. D. Lee, "A hybrid Bayesian network approach to detect driver cognitive distraction," *Trasp. Res. Part C: Emerging Technologies*, vol. 38, pp. 146–155, Jan. 2014.

[57] M. Kutila, M. Jokela, T. Mäkinen, J. Viitanen, G. Markkula, and T. Victor, "Driver cognitive distraction detection: Feature estimation and implementation," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 221, no. 9, pp. 1027–1040, 2007.

[58] N. Li and C. Busso, "Using perceptual evaluation to quantify cognitive and visual driver distractions," in *Smart Mobile In-Vehicle Systems*. Springer, 2014, pp. 183–207.

[59] F. Tango and M. Botta, "Real-Time Detection System of Driver Distraction Using Machine Learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 894–905, Jun. 2013.

[60] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[61] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *J. R. Stat. Soc. Ser. C Appl. Stat.*, pp. 100–108, 1979.

[62] V. Y. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem," *Neural Netw.*, vol. 4, no. 3, pp. 381–383, Jun. 1991.

[63] B. Mohar, "The laplacian spectrum of graphs," in *Graph theory, Combinatorics, and Applications*, 1991, pp. 871–898.

[64] K. Fan, "On a theorem of Weyl concerning eigenvalues of linear transformations I," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 35, no. 11, p. 652, 1949.

[65] A. Kessy, A. Lewin, and K. Strimmer, "Optimal whitening and decorrelation," *Am. Stat.*, vol. -, no. -, pp. –, 2017.

[66] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 4, 2007.

[67] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinformatics*, vol. 8, no. 1, p. 3, 2007.

[68] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recogn.*, vol. 41, no. 1, pp. 191–203, Jan. 2008.

[69] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. 20, no. 1, pp. 68–86, Jan. 1971.

[70] D. Cai, X. He, J. Han, and H.-J. Zhang, "Orthogonal Laplacianfaces for face recognition," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3608–3614, Nov. 2006.

[71] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge, MA, USA: MIT Press, 2001, pp. 849–856.

[72] Z. Xu, R. Jin, J. Zhu, I. King, M. Lyu, and Z. Yang, "Adaptive regularization for transductive support vector machine," in *Proc. 23rd Annu. Conf. NIPS*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds. Curran Associates, 2009, pp. 2125–2133.

[73] V. Garla, C. Taylor, and C. Brandt, "Semi-supervised clinical text classification with Laplacian SVMs: An application to cancer case management," *J. of Biomedical Informatics*, vol. 46, no. 5, pp. 869–875, Oct. 2013.

[74] V. Vapnik, *The nature of statistical learning theory*. New York: Springer, 2000.

[75] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press, 2001.

[76] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, pp. 1149–1184, 2011.

[77] Y. Yang, "The effects of increased workload on driving performance and visual behaviour," Ph.D. dissertation, Univ. of Southampton, Southampton, UK, Dec. 2011.

[78] *faceLAB 5 User Manual*, seeingmachines, Canberra, Australia, 2009.

[79] A. Duchowski, *Eye tracking methodology: Theory and practice*, 2nd ed. London, United Kingdom: Springer, 2007.

[80] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance," Federal Highway Admin., Washington, DC, Tech. Rep. FHWA-MCRT-98-006, 1998.

[81] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[82] Bayes Net Toolbox for Matlab. [Online]. Available: https://code.google.com/p/bnt/

[83] P. Leray and O. Francois, "BNT structure learning package: Documentation and experiments," Laboratoire PSI, INSA Rouen, Tech. Rep. FRE CNRS 2645, 2006.

[84] N. Zeidat, C. F. Eick, and Z. Zhao, "Supervised clustering: algorithms and applications," Dept. of Comput. Sci., Univ. Houston, Houston, TX, Tech. Rep. UH-CS-06-10, 2006.

[85] V. Sindhwani and S. S. Keerthi, "Large scale semi-supervised linear SVMs," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. and Develop. Inform. Retrieval*. ACM, 2006, pp. 477–484.

[86] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, 2007.

[87] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Syst. and Technol.*, vol. 2, no. 3, p. 27, May 2011.

[88] C. A. R. de Sousa, S. O. Rezende, and G. E. A. P. A. Batista, "Influence of Graph Construction on Semi-supervised Learning," in *Mach. Learn. and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 8190, pp. 160–175.

[89] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, 2006.

[90] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[91] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[92] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *Proc. ICML*, 2000.

[93] S. Akogul and M. Erisoglu, "An approach for determining the number of clusters in a model-based cluster analysis," *Entropy*, vol. 19, no. 9, p. 452, 2017.

[94] Q. Song, "A robust information clustering algorithm," *Neural Computation*, vol. 17, no. 12, pp. 2672–2698, 2005.