Universidad del País Vasco
Euskal Herriko Unibertsitatea

Department of Computer Science and Artificial Intelligence
Departamento de Ciencias de la Computación e Inteligencia Artificial

Graph-based Semi-supervised Learning: Algorithms and Applications

by

Libo Weng

Supervised by
Dr. Fadi Dornaika & Dr. Zhong Jin

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country (UPV/EHU) as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, May 2017

Acknowledgement

I would like to thank my supervisors, Dr. Fadi Dornaika and Dr. Zhong Jin, for their instructive suggestions and valuable comments on the writing of this research. Without their invaluable help and generous encouragement, this work would not have been accomplished.

Besides, I wish to thank all my friends in Spain and France, and mainly Alireza for his help during my stay in San Sebastian, and all the colleagues and friends in both UPV and UTBM and all the friends Igñacio, Luca, Paolin, etc. for their help and company in the residence LaSalle.

I also need to thank all my friends in Nanjing and all the members in Lab-403 and later Lab-4053.

Finally, I am grateful to my family for their patience and love and for their unconditional support.

Abstract

Graph-based semi-supervised learning have attracted large numbers of researchers and it is an important part of semi-supervised learning. Graph construction and semi-supervised embedding are two main steps in graph-based semi-supervised learning algorithms. In this thesis, we proposed two graph construction algorithms and two semi-supervised embedding algorithms. The main work of this thesis is summarized as follows:

1. A new graph construction algorithm named Graph construction based on self-representativeness and Laplacian smoothness (SRLS) and several variants are proposed. Researches show that the coefficients obtained by data representation algorithms reflect the similarity between data samples and can be considered as a measurement of the similarity. This kind of measurement can be used for the weights of the edges between data samples in graph construction. Each column of the coefficient matrix obtained by data self-representation algorithms can be regarded as a new representation of original data. The new representations should have common features as the original data samples. Thus, if two data samples are close to each other in the original space, the corresponding representations should be highly similar. This constraint is called Laplacian smoothness. SRLS graph is based on ℓ_2-norm minimized data self representation and Laplacian smoothness. Since the representation matrix obtained by ℓ_2 minimization is dense, a two phrase SRLS method (TPSRLS) is proposed to increase the sparsity of graph matrix. By extending the linear space to Hilbert space, two kernelized versions of SRLS are proposed. Besides, a direct solution to kernelized SRLS algorithm is also introduced.

2. A new sparse graph construction algorithm named Sparse graph with Laplacian smoothness (SGLS) and several variants are proposed. SGLS graph algorithm is based on sparse representation and use Laplacian smoothness as a constraint (SGLS). A kernelized version of the SGLS algorithm and a direct solution to kernelized SGLS algorithm are also proposed.

3. SPP is a successful unsupervised learning method. To extend SPP to a semi-supervised embedding method, we introduce the idea of in-class constraints in CGE into SPP and propose a new semi-supervised method for data embedding named Constrained Sparsity Preserving Embedding (CSPE).

4. The weakness of CSPE is that it can not handle the new coming samples which means a cascade regression should be performed after the non-linear mapping is obtained by CSPE over the whole training samples. Inspired by FME, we add a regression term in the objective function to obtain an approximate linear projection simultaneously when non-linear embedding is estimated and proposed Flexible Constrained Sparsity Preserving Embedding (FCSPE).

Extensive experiments on several datasets (including facial images, handwriting digits images and objects images) prove that the proposed algorithms can improve the state-of-the-art results.

Keywords: *Graph construction, Data self-representation, Laplacian smoothness, Sparse representation, Face recognition, Semi-supervised embedding, Dimensionality reduction*

Contents

List of Figures

List of Tables

1

Introduction

Contents

With the rapid development of computer technology and the widespread application of sensor technology, the data collected is growing exponentially. How to effectively utilize the information in these data to enhance the ability of knowledge representation and acquirement is one of the long-term goals of practitioners and researchers. The goal of machine learning is to discover and explore the interesting information hidden in the data, to learn the law from these data and to improve the system performance.

Machine learning is a core subject in the field of artificial intelligence and pattern recognition, and is categorized into unsupervised learning, supervised learning, and semi-supervised learning according to the use of sample labels. Learning from limited amounts of labeled data, also referred to as supervised learning, is one of the most popular research tasks in machine learning area. The labels of the data is usually prepared based on human effort, which is expensive to obtain, difficult to scale, and often error prone. At the same time, unlabeled data is readily available in large quantities in many domains. In order to benefit from such widely available unlabeled data, several semi-supervised learning (SSL) algorithms have been developed over the years. The semi-supervised learning algorithms use both labeled and unlabeled samples thus benefit from unlabeled as well as labeled data.

With the explosive growth of the Internet, graph structured datasets are becoming widespread and researchers have started to realize that graphs provide a natural way to represent data in a variety of different domains. In such datasets, nodes correspond to data samples, while edges represent relationships among nodes (samples).

Graph-based SSL techniques bring together these two lines of research. In particular, starting with the graph structure and label information about a subset of the nodes, graph-based SSL algorithms classify the remainder of the nodes in the graph. Graph-based SSL algorithms have been shown to outperform non graph-based SSL approaches [75]. Furthermore, majority of the graph-based SSL approaches can be optimized using convex optimization techniques and are both easily scalable and parallelizable. Graph-based SSL algorithms have been successfully used in many different applications areas.

In order to have a good understanding of graph-based SSL usefulness in machine learning, we will briefly go through unsupervised learning, supervised learning and semi-supervised learning.

Throughout the dissertation capital bold letters refer to matrices and small bold letters refer to vectors.

1.1 Unsupervised learning

The training data in the case of unsupervised learning contains no supervisory information. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \Re^{m \times n}$ be the data matrix, where n is the number of training samples and m is the dimension of each sample. It is assumed that each $\mathbf{x}_i$ is sampled independently. Thus, these samples are independent and identically distributed or i.i.d.

Examples of unsupervised learning problems include clustering and dimensionality reduction. The goal of clustering is to group similar samples in $\Re^m$, while dimensionality reduction aims to represent each sample with a lower dimensional feature vector with as little loss of information as possible.

Principal Component Analysis (PCA) [93] and Multidimensional Scaling (MDS) [8] are two classic unsupervised learning methods.

1.2 Supervised learning

In supervised learning, every sample in $\mathbf{X}$ has both the input $\mathbf{x} \in \Re^m$ and the corresponding label $y \in Y$. Here the label can refer to class membership or to one or more continuous variables. Formally, given a training set $(\mathbf{x}_i, y_i)_{i=1}^n$, the goal of a supervised learning algorithm is to train a function $f : \mathbf{X} \to Y$. When a trained model is obtained, $f^*(\mathbf{x})$, the prediction of a new sample $\mathbf{x}_{new}$ will be $y_{test} = f^*(\mathbf{x}_{test})$.

Based on Y, supervised learning can be categorized into classification and regression. When Y is discrete, the supervised learning tasks are referred to as classification while Y is continuous are called regression.

Linear discriminant analysis (LDA) [29, 2, 18] is a famous supervised linear dimensionality algorithm.

Supervised learning very often requires annotating large amounts of data. This can be a drawback since it requires extensive human intervention and

supervision. This can be both time consuming and often error prone. Unsupervised learning, on the other hand, requires no "labeled" training data but suffers from the inability for one to specify the expected output for a given input.

1.3 Semi-supervised learning

Semi-supervised learning (SSL) lies somewhere between supervised and unsupervised learning. It combines the advantages of both supervised and unsupervised learning. Here only a small amount of the training set $\mathbf{X}$ is labeled while a relatively large fraction of the training data is left unlabeled. The goal of a SSL algorithm is to learn a function $f : \mathbf{X} \to Y$, with a training set $\mathbf{X} = [\mathbf{X}_L, \mathbf{X}_U]$ where $\mathbf{X}_L$ is the data set with labels and $\mathbf{X}_U$ represents the set of unlabeled training samples.

However, it appears that the unlabeled data contain no information about this mapping. In general, SSL algorithms make one or more of the following assumptions so that information available in the unlabeled data can influence $f : \mathbf{X} \to Y$.

1. Smoothness Assumption: if two points in a high-density region are close then their corresponding outputs are also close.
2. Cluster Assumption: if two points are in the same cluster, they are likely to be of the same class. Another way to state this assumption would be to say that the decision boundary should lie in a low-density region.
3. Manifold Assumption: high-dimensional data lies within a low-dimensional manifold. This is very important owing to the fact that most machine learning algorithms suffer from the "curse of dimensionality". Thus, being able to handle the data on a relatively low-dimensional manifold can often be very advantageous for the algorithms.

One of the earliest SSL algorithm is self-training [68]. In many instances, expectation-maximization (EM) [24] can also be seen as an SSL algorithm. EM is a general procedure to maximize the likelihood of the data given a model with hidden variables and is guaranteed to converge to a local maximum. EM lends itself naturally to SSL as the labels for the unlabeled data can be treated as missing (hidden) variables. Example of algorithms that use EM for SSL include [37, 53, 58]. Co-training is another SSL algorithm [7] that is related to self-training and takes advantage of multiple views of the data. Transductive support vector machines (TSVM) [85] are based on the premise that the decision boundary must avoid high density regions in the input space. Other SSL algorithms can be found in [15].

1.4 Graph-based semi-supervised learning

Graph-based SSL algorithms are an important sub-class of SSL techniques that have received much attention in the recent past. Here, one assumes that the data (both labeled and unlabeled) is embedded within a low-dimensional manifold that may be reasonably expressed by a graph. Each data sample is represented by a vertex in a weighted graph with the weights providing a measure of similarity between sample vertices.

Thus, taking a graph-based approach to solving a SSL problem involves the following steps:

1. graph construction over the input data,
2. injecting seed labels on a subset of the nodes,
3. inferring labels on unlabeled nodes in the graph.

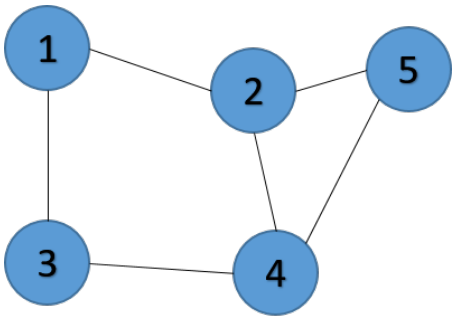


Fig. 1.1: Adjacency graph. The similarity score of connected nodes is 1, 0 for that of unconnected nodes.

The graph construction step is quite important in graph-based SSL algorithms. Figure 1.1 shows a sample of a simple graph in which the similarity score is binary (1 if two nodes are connected, 0 otherwise). The affinity matrix of the graph in Figure 1.1 is given by:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Thus, graph construction is an important step in graph-based SSL algorithms. Compared to the normal semi-supervised learning, graph based SSL methods has the following advantages over other approaches [76]:

1. For many applications, graph based SSL performs better than most other SSL algorithms in comparative evaluations.

2. Most graph based methods have a convex objective function providing convergence guarantees, making them attractive for solving large-scale problems.
3. For many graph-based SSL approaches, optimizing the objective can be achieved via message passing on graphs. Each iteration of the algorithm consists of a set of updates to each graph node. A node's updated value is computed based on the node's current value as well as on the neighbors' current set of values.
4. It is possible to derive simple fast heuristics that enable such algorithms to scale to large parallel machines with good machine efficiency.

1.5 Thesis organization

The dissertation is structured in three parts.

- Part I, *Graph construction* which focuses on the problem of graph construction.
  It is composed of three chapters:
  - Chapter 3 reviews some state-of-the-art graph construction methods.
  - Chapter 4 describes the proposed graph construction method based on data self-representativeness and Laplacian smoothness.
  - Chapter 5 introduces a new sparse graph construction method using Laplacian smoothness as a constraint.
- Part II, *Semi-supervised manifold learning* describes several approaches developed for semi-supervised embedding.
  Two chapters are deployed:
  - Chapter 6 introduces the semi-supervised manifold learning concept and describes different algorithms related.
  - Chapter 7 presents the theory of our proposed Constrained Sparsity Preserving Embedding and Flexible Constrained Sparsity Preserving Embedding methods. Then the experiments on several datasets are presented to evaluate the proposed methods and compare its performance with other manifold learning techniques.
- Part III
  - Chapter 8 summarizes and concludes the developed work and discusses the advantages and limitations of the proposed methods. It also gives some directions for future research.

2

Datasets and experimental setup

Contents

In our thesis, nine real datasets are used including face images, object images, handwritten images and text datasets.

2.1 Face datasets

The following seven face datasets are used for experimental evaluations in the thesis.

- **Yale:**[1] This database contains 165 face images from 15 people and each person has 11 images. The images are resized to 32×32 for processing.
- **ORL:**[2] This database contains 400 face images from 40 people and each person has 10 images. The size of the original image is 92×112. We resize the images to 32×32 for processing.
- **FERET:**[3] The subset used contains 1400 face images of 20 individuals and each person has 7 images. The images are resized to 32×32 for processing.
- **PF01:** This database contains 1751 images from 17 individuals. The images are resized to 32×32 for processing.
- **PIE:**[4] This database contains 1926 images from 68 individuals. The images are resized to 32×32 for processing.

[1] *http://vision.ucsd.edu/content/yale-face-database*
[2] *http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html*
[3] *http://www.nist.gov/itl/iad/ig/colorferet.cfm*
[4] *http://www.ri.cmu.edu/projects/project_418.html*

- **Extended Yale B:**[5] The subset used contains 1680 face images of 28 individuals. The images are resized to 32×32 for processing.
- **LFW:**[6] The LFW (Labeled Faces in the Wild) data set is designed for unconstrained face verification tasks. Images of this data set are collected from the web. The original version contains more than 13000 images. The face in this data set is misaligned. Our experiments are made on a set of 1551 images referring to 141 subjects (11 images for each subject). This selection of images is motivated by the fact that LFW data set is designed for face verification problems rather than face identification in our case.
- **LFW-a:**[7] The LFW-a data set is the aligned version of LFW data set. Our experiments are made on a set of 1551 images referring to 141 subjects (11 images for each subject).

2.2 Object dataset

- **COIL-20:**[8] A subset of this database contains 360 images of 20 objects with different rotations. The square root of Local Binary Patterns (LBP) feature is used for processing [98].

2.3 Handwritten dataset

- **USPS:**[9] A subset of this database contains 1100 images of 10 handwritten digits. The images are of size 16×16 pixels.

Figure 2.1 shows image samples from the above nine datasets.

[5] *http://vision.ucsd.edu/∼leekc/ExtYaleDatabase/ExtYaleB.html*
[6] *http://vis-www.cs.umass.edu/lfw*
[7] *http://www.openu.ac.il/home/hassner/data/lfwa/*
[8] *http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php*
[9] *http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html*

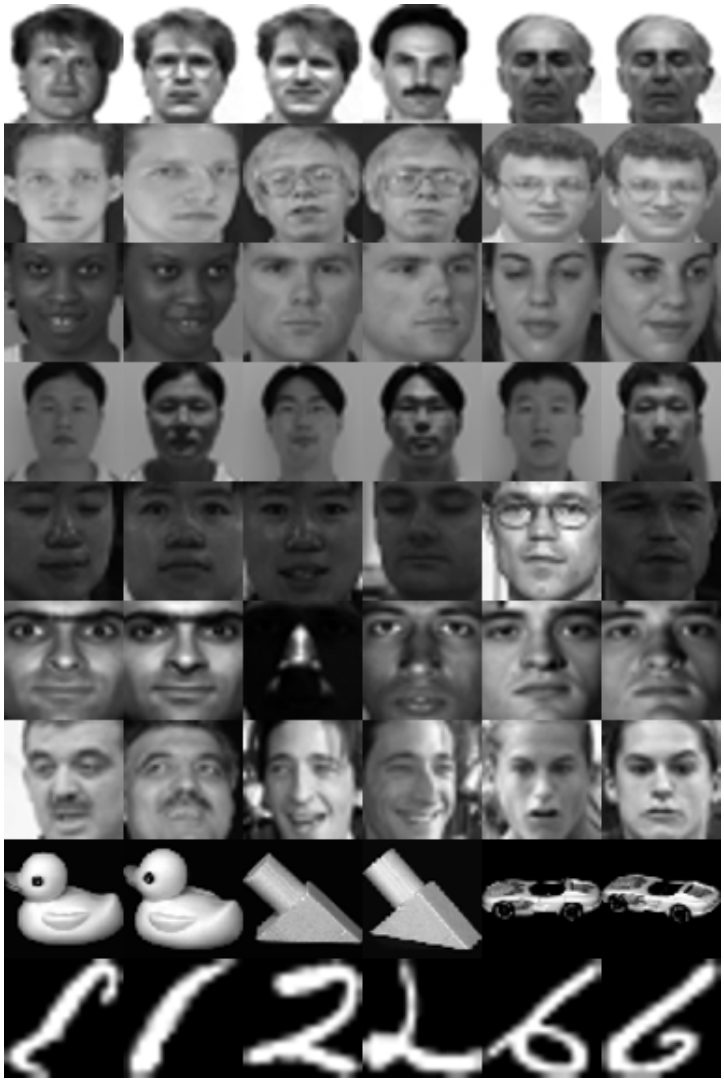


Fig. 2.1: Six image samples are shown for the nine datasets.

Part I

Graph Construction

3

Advances in graph construction

Abstract

In this chapter we will have a brief introduction about graph construction and then review the existing graph construction methods. And then we introduce the label propagation methods which can be used to evaluate the constructed graphs by methods.

Contents

3.1 Introduction

Most real world data can have a graph structure that describes the pairwise similarity or relationship among samples. It is realized that data graphs are a natural way to represent the data [63, 26, 87, 104, 96]. Graph-based methods operate on a data driven graph [112, 41, 88, 91, 94, 65, 81, 40]. In the graphs, the nodes correspond to data samples and the weighted edges between nodes encode the similarity between two nodes.

The most common way to construct a graph, or equivalently to estimate its affinity matrix, is to construct k-nearest neighbor graphs [5] or ε-neighborhoods graphs. Then edge weights are estimated using a similarity function that quantifies the relationship between the sample and its neighbors. It is noticed that the parameter setting in these two kinds of methods will heavily influence the final task performance [23]. Indeed, there is no simple way that can predict the best parameter setting for these methods. Jebara et

al. [41] propose a graph construction method by b-matching in order to force that all the nodes will have the same degree (degree means the number of the edges connected to the node). In the past decade, different coding schemes and code book generation methods have been proposed.

In [22], the authors aim at constructing hard graphs using a similar criterion used by [66]. In this work, the neighborhood selection and edge weighting are performed in a single step; the edge weight matrix is symmetric and only non-negative edge weights are allowed. The edge weights are computed using a constraint that forces the degree, or the weighted degree of every node, to be equal to or greater than one. The authors devise a quadratic program that computes the non-negative weights. In order to avoid the non-tractability of solution, the graph is incrementally constructed by solving quadratic programs with a subset of edges.

Wang et al. [88] use a similar criterion as Locally Linear Embedding (LLE) to construct the graph by calculating the weights between pairs of samples. Wei et al. [91] define a neighborhood preserving graph based on LLE criterion for semi-supervised dimensionality reduction. Sparse representation is a widely used technique which assumes the complex signal can be represented by some basic signals. Researchers start to be interested in sparse graphs since they provide good performance in several post-graph learning tasks. it was found that sparsity can be obtained by introducing ℓ_1 regularization on the unknown coefficients [94]. Qiao et al. [65] construct an ℓ_1 graph with weighted edges using the coefficients of the sparse coding based on the theory of sparse representation and apply it to a locality preserving projection method for human face recognition. Yan et al. [97] also use the coefficients of sparse representation to construct the graph for semi-supervised classification [20] and multi-label classification [86].

Moreover, different from the sparse representation, [90] proposed collaborative neighborhood representation (CNR) based on ℓ_2 norm minimization. The proposed coding was used as a variant to the Sparse Representation Classifier [94]. Thus, graph construction using the coefficients of CNR is also a good choice. Locality-constrained Linear Coding (LLC) [89] is another algorithm for data representation. Based on CNR and LLC, Dornaika et al. proposed a graph construction method named weighted regularized least square [26] and a two phase method in paper [27].

As mentioned previously, the two main steps of a graph-based SSL problem are: 1. constructing a graph over the data samples; 2. using appropriate learning algorithm to infer the unlabeled samples in the constructed graph.

3.2 Overview of the existing graph construction methods

The graph-based SSL method acts on a graph in which a node represents a data sample and the connected edge between the nodes is given a weight.

In the real world, the relationship between some data is naturally a well-structured graph, but for most learning tasks it is necessary to assume that the data samples are independent and identically distributed, and need to construct a graph that is suitable for graph-based SSL tasks.

Consider a set of n data samples (measurements, signals or images), taken from a m dimensional space, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. This set is usually represented by a $m \times n$ matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \Re^{m \times n}$. A graph can be represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \mathbf{W})$, where $\mathcal{V}$ is the set of nodes ($|\mathcal{V}| = n$), every node represents a data sample, $\mathcal{E}$ is the set of edges and $\mathbf{W}$ is the $n \times n$ edge weight matrix which is also called the affinity matrix. The weight of the edge $(\mathbf{x}_i, \mathbf{x}_j)$ is given by W_{ij} which quantifies the similarity between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$.

3.2.1 Traditional graph construction methods

k-nearest neighbor graphs and ε-neighborhood graphs are two traditional graph construction methods.

kNN graphs: Samples $\mathbf{x}_i$, $\mathbf{x}_j$ are connected by an edge if $\mathbf{x}_i$ is in $\mathbf{x}_j$'s k-nearest neighborhood or vice versa. k is a parameter that controls the density of the graph.

ε-neighborhood graphs: Samples $\mathbf{x}_i$, $\mathbf{x}_j$ are connected by an edge if the distance $d(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon$. The parameter ε controls neighborhood radius.

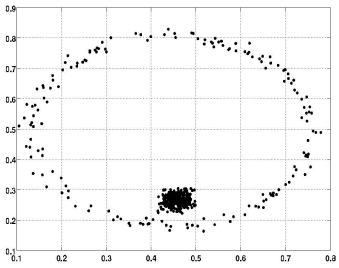
After the edges are decided, similarity will be measured to weight the edges. The formula is shown as Eq.(3.1). For instance, $sim(\mathbf{x}_i, \mathbf{x}_j)$ is set to 1 or to $e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$, where σ is a parameter.

$$W_{ij} = \begin{cases} sim(\mathbf{x}_i, \mathbf{x}_j), & \mathbf{x}_i \in \delta_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \delta_k(\mathbf{x}_i), \\ 0, & \text{otherwise}, \end{cases} \tag{3.1}$$

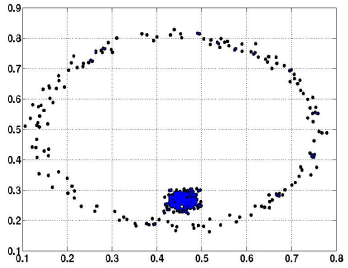
where $\delta_k(\mathbf{x}_i)$ represents the set of $\mathbf{x}_i$'s k-nearest neighbors.

Lots of work show that the choice of parameter k or ε can significantly affect the results. Figure 3.1 shows kNN and ε-neighborhood graphs constructed from a synthetic dataset [41]. We can see that the ε-neighborhood method is quite sensitive to the choice of ε and it may return graphs with disconnected components.

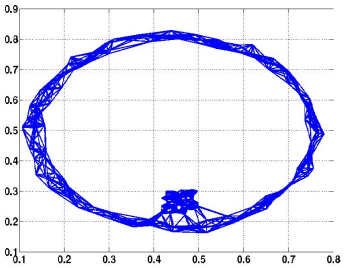
Since kNN graphs and ε-neighborhood graphs can not promise the degree of all the nodes are the same, the authors propose an adjacency graph construction via b-matching in [41] which can produce an undirected adjacency graph matrix with the constraint that all the nodes in the graph will have the same degree given by the parameter b. The solution was obtained by loopy belief propagation. It was shown that the label propagation algorithm that uses the resulting adjacency graph has better performance than that based on kNN graph. However, the b-matching graph construction needs tuning the parameter b.



(a) The synthetic data



(b) ε-neighborhood graph



(c) kNN graph

Fig. 3.1: kNN and ε-neighborhood graphs constructed from a synthetic dataset.

3.2.2 Locally Linear Embedding based graph construction

Locally Linear Embedding (LLE) is a classic manifold learning method. LLE preserves the neighborhood relationships of input samples [66]. It exploits the local linear reconstructions by minimizing the reconstruction error of the set of all local neighborhoods in the input space. It is discovered that the linear coding used by LLE can be used for the graph weight matrix construction.

The affinity matrix $\mathbf{W}$ can be obtained by minimizing the reconstruction error: For each original data $\mathbf{x}_i$, the corresponding representation coefficient vector can be obtained by minimizing the following reconstruction error:

$$\begin{aligned} \min \ & \sum_{i=1}^{n} \|\mathbf{x}_i - \Sigma_{\mathbf{x}_j \in \delta_k(\mathbf{x}_i)} W_{ij} \mathbf{x}_j\|^2, \\ s.t. \ & \sum_{j=1}^{n} W_{ij} = 1, \end{aligned} \tag{3.2}$$

where $\delta_k(\mathbf{x}_i)$ represents the set of $\mathbf{x}_i$'s k-nearest neighbors.

3.2.3 Graph construction via ℓ_2 minimization

3.2.3.1 Standard ℓ_2 graph

Data representation methods provide coefficients between one sample and the rest samples. Normally two similar samples will have large coefficient in representation. Thus the coefficients obtained by representation methods can be used for graph affinity matrix.

In paper [105], a representation method named collaborative representation is proposed. It is a standard ℓ_2 coding based representation method which is to solve an ℓ_2 norm problem:

$$\mathbf{w}_i = \arg\min_{\mathbf{W}_i} \|\mathbf{x}_i - \mathbf{X}\,\mathbf{w}_i\|_2^2 + \lambda\|\mathbf{w}_i\|_2^2, \tag{3.3}$$

where the vector $\mathbf{w}_i$ is the set of coefficients W_{ij} depicting the relation between the sample $\mathbf{x}_i$ and the samples $\mathbf{x}_j$.

The coefficients obtained above for each data sample can form the weight matrix of the graph. The solution to $\mathbf{w}_i$ is given by

$$\mathbf{w}_i = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{x}_i. \tag{3.4}$$

Thus, the affinity matrix $\mathbf{W}$ consists of all the coefficient vectors related to each samples:

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n], \tag{3.5}$$

where, $\mathbf{w}_i$ is given by (3.4).

3.2.3.2 Weighted regularized least square minimization

The weighted regularized least square (WRLS) [26] is based on a weighted ℓ_2 regularizer that incorporates the locality of one sample by exploiting its distances to the rest of samples. In WRLS method, the coding vector of sample $\mathbf{x}_i$ (i.e., the i^{th} vector of the unknown affinity matrix $\mathbf{W}$) is estimated by minimizing the following criterion:

$$\mathbf{w}_i = \arg\min_{\mathbf{W}_i} \|\mathbf{x}_i - \mathbf{X}\,\mathbf{w}_i\|_2^2 + \lambda\sum_{j=1}^{n}\left(P_{jj}^{(i)}\,w_{ij}\right)^2, \tag{3.6}$$

where $\mathbf{w}_i = [w_{i1}, \ldots, w_{i,i-1}, 0, w_{i,i+1}, \ldots, w_{in}]^T$ and $P_{jj}^{(i)} = 1 - e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}$.

The solution is given by:

$$\mathbf{w}_i = \left(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{P}^{(i)}\right)^{-1}\mathbf{X}^T\mathbf{x}_i, \tag{3.7}$$

where $\mathbf{P}^{(i)}$ is a diagonal matrix with $P_{jj}^{(i)}$.

When $P_{jj}^{(i)} = 1$, the problem will be reduced to the standard ℓ_2 coding Eq.(3.3).

3.2.4 Sparse graph construction via ℓ_1 minimization

3.2.4.1 Standard sparse graph

The traditional graphs need to choose some parameters in advance (e.g., the neighborhood size k and the Gaussian parameter σ), while sparsity representation based graph is parameter-free. [65] and [97] proposed sparsity representation based graph construction methods in which every sample is represented as a sparse linear combination of the rest of input samples and the coefficients are considered as weights.

$$\begin{aligned} &\min \|\mathbf{w}_i\|_1, \\ &s.t.\ \mathbf{x}_i = \mathbf{X}\,\mathbf{w}_i, \end{aligned} \tag{3.8}$$

where $\mathbf{w}_i = [w_{i1}, \ldots, w_{i,i-1}, 0, w_{i,i+1}, \ldots, w_{in}]^T$ is an n-dimensional vector in which the i-th element is equal to zero (implying that the $\mathbf{x}_i$ is removed from $\mathbf{X}$), and the elements $w_{ij}, j \neq i$ denote the contribution of each $\mathbf{x}_j$ to reconstructing $\mathbf{x}_i$.

After the weight vector $\mathbf{w}_i$ for each $\mathbf{x}_i$, $i = 1, 2, \ldots, n$ is obtained, the sparse reconstructive weight matrix $\mathbf{W} = (w_{ij})_{n\times n}$ can be defined as follows:

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]^T, \tag{3.9}$$

where $\mathbf{w}_i$ is the optimal solution of Eq.(3.8).

A robust version of sparse reconstructive graph is to solve the following ℓ_1 problem:

$$\begin{aligned} &\min \|\mathbf{w}_i\|_1 + \|\mathbf{e}\|_1, \\ &s.t.\ \mathbf{x}_i = \mathbf{X}\,\mathbf{w}_i + \mathbf{e}. \end{aligned} \tag{3.10}$$

The problem above can be rewritten as:

$$\begin{aligned} &\min \| \left[\mathbf{w}_i^T\ \mathbf{e}^T\right]^T \|_1, \\ &s.t.\ \mathbf{x}_i = \left[\mathbf{X}\ \mathbf{1}\right] \begin{bmatrix} \mathbf{w}_i \\ \mathbf{e}_i \end{bmatrix}. \end{aligned} \tag{3.11}$$

In the sequel, the graph obtained with the coding of Eq.(3.11) is called robust sparse graph (ℓ_1-r).

3.2.4.2 Constrained sparse graph

In [19], the authors propose a constrained sparse graph using a variant of sparse graph (linear reconstruction with ℓ_1 norm regularization). Their method tries to estimate the unknown graph affinity matrix by minimizing the following criterion:

$$\begin{aligned} &\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{X}\,\mathbf{W}\|_F^2 + \alpha \|\mathbf{W} - \mathbf{S}\|_F^2 + \beta \|\mathbf{W}\|_1, \\ &s.t.\ \mathbf{W}_{ij} = \mathbf{W}_{ji} \geq 0, \mathbf{W}_{ii} = 0, \end{aligned} \tag{3.12}$$

where $\mathbf{S}$ is a known matrix that can quantify the pairwise similarity of data, and parameters $\alpha > 0, \beta > 0$ are two balance parameters. The second term in Eq.(3.12) can be seen as a constraint on the sought affinity matrix. The authors deploy an algorithm that tries to iteratively update a predefined similarity matrix until the objective criterion changes less than a threshold. As can be seen, the algorithm can inherit the advantages of sparse graphs. However, it needs a reference similarity matrix (the matrix $\mathbf{S}$) that can be affected by the parameter setting.

In the sequel, graph obtained with the coding of Eq.(3.12) is called constrained sparse graph (ℓ_1-c).

3.3 Learning and Inference

Graph-based learning tasks are numerous. These include label propagation, regression, spectral clustering, manifold learning, dimensionality reduction, and feature selection. In the section, we focus on label propagation since it is a key solution to many semi-supervised problems.

Once the graph is constructed, the next step in solving an SSL problem using graph-based methods is the injection of seed labels on a subset of the nodes in the graph followed by the process of inferring the labels for the unlabeled nodes.

Label propagation algorithms can be very appealing since they learn from limited amounts of labeled data combined with widely available unlabeled data. Among the current SSL methods, graph based approaches have emerged as methods of choice for general semi-supervised tasks in terms of accuracy and computational efficiency. Most of the graph based SSL algorithms concentrate primarily on the label inference part, i.e. assigning labels to nodes once the graph is constructed.

Let's assume a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l, \mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}]$ corresponding to C classes, where l is the number of labeled samples and u is the number of unlabeled ones. The total number of data samples is $n = l + u$. Denote $\mathbf{X}_L = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l]$ and $\mathbf{X}_U = [\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}]$.

Let $\mathbf{W}$ denote the affinity matrix obtained by applying a given graph construction method on the data. Let the label indicator matrix for the labeled samples denoted by $\mathbf{Y}_L = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_l]^T \in \Re^{l \times C}$, where $y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j, 0 otherwise. The task is to estimate the label indicator matrix $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]^T \in \Re^{n \times C}$ so that the label of unlabeled samples can be inferred.

There are several label propagation algorithms (also called classifiers): Gaussian Random Fields (GRF) [114], Local and Global Consistency (LGC) [111], Robust multi-class Graph Transduction (RMGT) [51]. Recently a new label propagation method is proposed coined Label Prediction via Deformed Laplacian (LPDGL) [32] which can be regarded as an extension of LGC. The

author of LPDGL method claimed that their label propagation method can out perform the other existing label propagation algorithms.

Moreover, Baluja et al. proposed Adsorption [3] which can be regarded as a general framework for transductive learning. Talukdar and Crammer proposed MAD (Modified Adsorption) [80] based on this. The objective used in Quadratic Criteria (QC) [15] is similar to that in MAD. Orbach and Crammer proposed transduction with Confidence (TACO) [61]. The other algorithms can be found in [75, 51, 52].

The above SSL algorithms are transductive [16], which means that the algorithms can not be used directly for new samples. The algorithms which can be applied directly on new unseen data instances are called inductive methods. The above trasductive methods can be suitable for multiclass classification tasks, while inductive methods are normally for binary classification. Manifold Regularization [6, 72, 73, 59] is the representative example of the inductive methods. Laplacian Regularized Least Squares (LapRLS) [**?**] and Laplacian Support Vector Machine (LapSVM) [**?**] are two popular variants of the manifold regularization.

3.3.1 Gaussian Random Fields

The GRF algorithm estimates the unknown labels by solving the following optimization problem:

$$\begin{aligned} &\min \ \mathrm{Tr}(\mathbf{F}^T\mathbf{L}\mathbf{F}), \\ &s.t. \ \mathbf{F}_L = \mathbf{Y}_L, \end{aligned} \tag{3.13}$$

where $\mathbf{F}_L = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_l]^T$, $\mathbf{L}$ is the graph Laplaian and is given by $\mathbf{L} = (\mathbf{D}_r - \mathbf{W}) + (\mathbf{D}_c - \mathbf{W}^T)$, $\mathbf{D}_r$ and $\mathbf{D}_c$ are diagonal matrices whose elements are the row and column sums of the matrix $\mathbf{W}$, respectively.

The solution can be obtained by:

$$\mathbf{F}_U = -\mathbf{L}_{UU}^{-1}\mathbf{L}_{UL}\mathbf{Y}_L, \tag{3.14}$$

where $\mathbf{F}_U = [\mathbf{f}_{l+1}, \mathbf{f}_{l+2}, \ldots, \mathbf{f}_n]^T$, and $\mathbf{L} = \begin{bmatrix} \mathbf{L}_{LL} & \mathbf{L}_{LU} \\ \mathbf{L}_{UL} & \mathbf{L}_{UU} \end{bmatrix}$.

3.3.2 Local and Global Consistency

The LGC algorithm infers the labels by minimizing the following criterion:

$$\frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}\|\frac{1}{\sqrt{D_{ii}}}\mathbf{f}_i - \frac{1}{\sqrt{D_{jj}}}\mathbf{f}_j\|^2 + \mu\sum_{i=1}^{l}\|\mathbf{f}_i - \mathbf{y}_i\|^2\right), \tag{3.15}$$

where $\mu > 0$ is a balance parameter.

The solution to Eq.(3.15) is:

$$\mathbf{F} = \frac{\mu}{1+\mu}(\mathbf{I} - \frac{1}{1+\mu}\tilde{\mathbf{W}})^{-1}\mathbf{Y}, \tag{3.16}$$

where $\tilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ and $\mathbf{I}$ is an $n \times n$ identity matrix.

3.3.3 Label prediction via deformed graph Laplacian

Let $\mathbf{W}$ denote the affinity matrix obtained by applying a given graph construction method on the data. Let the label indicator matrix for the labeled samples denoted by $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]^T \in \Re^{n \times C}$, where $y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j, $y_{ij} = -1$ if $\mathbf{x}_i$ does not belong to class j, 0 if $\mathbf{x}_i$ is unlabeled. The task is to estimate the label indicator matrix $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]^T \in \Re^{n \times C}$ so that the label of unlabeled samples can be inferred.

The LPDGL algorithm estimates the unknown labels by solving the following optimization problem:

$$\min_{\mathbf{F}} \frac{1}{2}\{\beta \mathrm{Tr}(\mathbf{F}^T \mathbf{L} \mathbf{F}) + \gamma \mathrm{Tr}(\mathbf{F}^T (\mathbf{I} - \mathbf{D}/v)\mathbf{F}) + \|\mathbf{U}(\mathbf{F} - \mathbf{Y})\|_F^2\}, \tag{3.17}$$

where $\mathbf{L}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $\mathbf{D}$ is diagonal matrices with $D_{ii} = \sum_j W_{ij}$, $v = \sum_i D_{ii}$, $\mathbf{U}_{n \times n}$ is a diagonal matrix with $J_{ii} = 1$ if $\mathbf{x}_i$ is labeled and 0 otherwise. β and γ are two parameters.

The solution can be obtained by:

$$\mathbf{F} = [\mathbf{U} + \beta \mathbf{L} + \gamma(\mathbf{I} - \mathbf{D}/v)]^{-1} \mathbf{U} \mathbf{Y}. \tag{3.18}$$

3.4 Conclusion

In this chapter, we described several existing graph construction techniques and label propagation methods as semi-supervised classifiers.

4

Graph Construction Based on Data Self-representativeness and Laplacian Smoothness

Abstract

In this chapter, we will present our proposed method based on data self-representativeness and Laplacian smoothness (SRLS). Then we will introduce three variants of it. The first variant is given by a two phase SRLS method (TPSRLS). The remaining variants are given by two kernelized versions of SRLS. In addition to these three variants, we also propose a direct solution to SRLS that is able to handle the complexity of the Laplacian smoothness term that is cubic in the graph coefficients; The chapter includes experimental evaluation of the graph when applied to the task of graph-based semi-supervised learning.

Contents

4.1 Introduction

In essence, the traditional graph construction methods including kNN graph, ε-neighborhood graph and LLE graph are in two processes: adjacency construction based on the neighbors of each sample and weight calculation.

The LLE graph is based on finding a linear combination for each sample of its neighbors. The combination coefficients are are used as a measurement

of the weights between two samples. In paper [97], the authors point out that the representation coefficients obtained by data representation methods based on reconstruction error minimization affect the similarity between samples. Thus, these coefficients can be used as a measurement of the weights over the edges between samples. In this way, graph construction based on data representation can construct the graph in one single process by solving a minimization problem.

ℓ_2-norm minimization based data representation methods are common data representation methods. Since the regularizer of these methods is quadratic term, the closed-form solution to the problem can be obtained. At he same time, the ℓ_2-norm based methods are normally in low computational complexity and easy to implement. Lots of ℓ_2-norm minimization based data representation methods have been proposed [105, 71, 90].

4.2 Graph construction based on data self-representativeness and Laplacian smoothness

In classic self-representation problems, the goal is to obtain a representation matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$, which is the solution to the following regularized problem:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda \|\mathbf{B}\|_F^2. \tag{4.1}$$

The norm $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. $\mathbf{B}$ can be regarded as the asymmetric weight matrix associated to the training samples. In other words, $\mathbf{B}$ is the coding matrix obtained when the dictionary is set to the data themselves. For every sample $\mathbf{x}_i$, $\mathbf{b}_i$ is the coding vector needed to construct the sample $\mathbf{x}_i$, from the whole data set of samples.

In ideal cases, the coding vectors of similar original samples should also be similar. For instance, if $\mathbf{x}_i$ is very similar to $\mathbf{x}_j$, then the corresponding coding vectors $\mathbf{b}_i$ and $\mathbf{b}_j$ should be close to each other. Mathematically, this property is known by Laplacian smoothness criterion. Figure 4.1 is a visual illustration for principle of Laplacian smoothness.

For all data pairs, this criterion is given by:

$$\sum_{i,j} \|\mathbf{b}_i - \mathbf{b}_j\|^2 W_{ij} = \mathrm{Tr}(\mathbf{B}\,(\mathbf{D} - \mathbf{W})\,\mathbf{B}^T) = \mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T), \tag{4.2}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix; $\mathbf{L}$ is the graph Laplacian of the weight matrix $\mathbf{W}$, i.e. $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$ and $\mathbf{W}$ is a given graph.

We propose to construct an affinity matrix $\mathbf{B}$ that simultaneously uses data self-representation and Laplacian smoothness. Thus, our coding matrix will be the solution to the following minimization problem:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda \|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_{\mathbf{B}}\,\mathbf{B}^T), \tag{4.3}$$

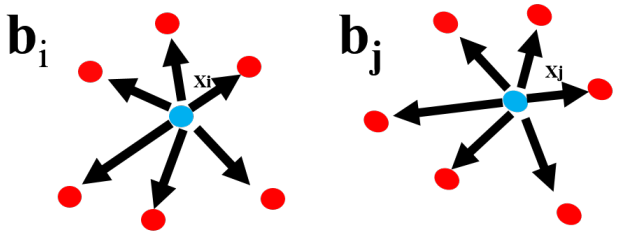


Fig. 4.1: Laplacian smoothness criterion (if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, then $\mathbf{b}_i$ and $\mathbf{b}_j$ should be similar).

where $\mathbf{L_B}$ is the Laplacian matrix of the affinity matrix $\mathbf{B}$, and λ and ρ are two positive balance parameters.

The proposed method (SRLS) is to solve the problem depicted in Eq.(4.3). Notice that if $\mathbf{L_B}$ is fixed, the functional in the Eq.(4.3) will be convex and have a closed form solution with a global minimum. However, the graph Laplacian is unknown since the graph itself is unknown. It can be seen that the optimization criterion in (4.3) poses a chicken-and-egg problem because the Laplacian matrix $\mathbf{L_B}$ needs to be known for computing $\mathbf{B}$. To solve that, we propose a new recursive method to get the optimal coding matrix (equivalently the graph). The intuition behind our idea is as follows. If we start with a rough graph, its corresponding Laplacian will impose smoothness of the vectors of the unknown coding matrix $\mathbf{B}$. Solving for $\mathbf{B}$ using the optimization problem in Eq.(4.3) with a fixed Laplacian matrix, the obtained solution satisfies data self-representation, regularized norm, and the Laplacian smoothness. The derived symmetric graph for the current estimated $\mathbf{B}$ will be better graph (in the sense of Laplacian smoothness). Therefore, successive minimizations having the form of Eq.(4.3) will improve the estimation of the coding matrix from which the final graph is derived. Note that in each minimization the current estimated coding matrix is used to generate the Laplacian matrix in the next minimization. In practice, we find that three to four successive minimizations of Eq.(4.3)) are enough to get a stable graph. This will be demonstrated in the experimental section.

The algorithm proceeds as follows. Firstly, a rough weight matrix $\mathbf{W}$ is computed by using any traditional graph construction method[1]. The graph Laplacian is then calculated by $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

Secondly, the matrix $\mathbf{B}$ is estimated by minimizing the following criterion:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T). \tag{4.4}$$

Let $Q(\mathbf{B}) = \|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T)$, we have

[1] The experimental section will show that even random graphs can also be used as initial graphs.

$$\begin{aligned}\frac{\partial Q(\mathbf{B})}{\partial \mathbf{B}} &= -2\mathbf{X}^T\mathbf{X} + 2\mathbf{X}^T\mathbf{X}\,\mathbf{B} + 2\lambda\,\mathbf{B} + 2\rho\,\mathbf{B}\,\mathbf{L} \\ &= 2(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I})\mathbf{B} + 2\rho\,\mathbf{B}\,\mathbf{L} - 2\mathbf{X}^T\mathbf{X}.\end{aligned} \tag{4.5}$$

The derivatives of $Q(\mathbf{B})$ w.r.t. the $\mathbf{B}$ should equal to $\mathbf{0}$ when $Q(\mathbf{B})$ reaches the minimum. $\frac{\partial Q(\mathbf{B})}{\partial \mathbf{B}} = \mathbf{0}$ yields:

$$(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I})\mathbf{B} + \rho\,\mathbf{B}\,\mathbf{L} = \mathbf{X}^T\mathbf{X}. \tag{4.6}$$

The above has the form of a Sylvester equation ($\mathbf{Y}\,\mathbf{B} + \mathbf{B}\,\mathbf{Z} = \mathbf{C}$). By solving this equation, we get the matrix $\mathbf{B}$.

Thirdly, $\mathbf{W}$ is obtained by $\mathbf{W} = (|\mathbf{B}| + |\mathbf{B}^T|)/2$.

The last two steps are repeated until $\mathbf{B}$ does not change. Thus, the procedure of SRLS is summarized in Algorithm 1.

Input: The original data matrix $\mathbf{X}$, a given rough weight matrix $\mathbf{W}_0$, λ, and ρ
Output: SRLS graph $\mathbf{W}$

Set $t = 0$, $\mathbf{W}_0$, ϵ (a small positive threshold);
repeat
  Calculate the graph Laplacian of $\mathbf{W}_t$, i.e. $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$;
  Get $\mathbf{B}_{t+1}$ by solving the following Sylvester equation
  $(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I})\mathbf{B} + \rho\,\mathbf{B}\,\mathbf{L}_t = \mathbf{X}^T\mathbf{X}$;
  Calculate new $\mathbf{W}_{t+1} = (|\mathbf{B}_{t+1}| + |\mathbf{B}_{t+1}^T|)/2$;
  $t = t + 1$;
until $\|\mathbf{W}_t - \mathbf{W}_{t-1}\|_F^2/n^2 < \epsilon$;
Get final SRLS graph $\mathbf{W} = \mathbf{W}_t$.

Algorithm 1: SRLS graph construction

It should be noticed that by setting the parameter ρ to zero, we get a graph construction method that is based on data self-representativeness only. In the sequel, we call this method ℓ_2 graph method. Indeed, if $\rho = 0$, the problem depicted in Eq.(4.3) reduces to n problems having exactly the form of Eq.(3.3).

Several iterative schemes to solve Sylvester equations have been proposed; for methods focusing on large sparse systems one can use the method described in [12].

4.3 Two phase SRLS (TPSRLS)

The proposed SRLS method is mainly based on data self-representativeness and Laplacian smoothness. In order to incorporate a kind of sparsity in the

final graph, we invoke a second phase of coding aiming at pruning many samples and estimating new coding coefficients.

Let $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$ be the optimal graph affinity obtained by the S-RLS method mentioned above. Let $\mathbf{W}$ be the corresponding symmetric affinity matrix. The row vector (or column vector) $\mathbf{w}_i$ contains a lot of elements that are small. This means that the corresponding samples and $\mathbf{x}_i$ are not highly similar. To make the final graph sparser, we remove the edges in the graph with small weight (i.e., we vanish these weights) and perform a new coding over the remaining samples. It is reasonable to set an adaptive threshold to eliminate small weights in each $\mathbf{w}_i$. The threshold function can be set as

$$Th(\mathbf{w}_i) = STAT(w_{i1}, w_{i2}, \ldots, w_{in}), \tag{4.7}$$

where $STAT(w_{i1}, w_{i2}, \ldots, w_{in})$ is a statistical function that can be set as a threshold. One simple choice is the mean, i.e. $Th(\mathbf{z}) = \frac{1}{n}\sum_{j=1}^{n} z_j$, where $\mathbf{z}$ is any n-dimensional vector.

Let R_i denote the set of samples whose W_{ij} is greater than the chosen threshold $Th(\mathbf{w}_i)$, i.e., $R_i = \{\mathbf{x}_j \mid W_{ij} > Th(\mathbf{w}_i)\} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_s}\}$ where n_s is the cardinality of R_i. Let $\mathbf{X}_s^{(i)}$ be the $d \times n_s$ data matrix associated with R_i. Then, the new coding vector $\mathbf{w}_i'$ associated with the selected examples will be solved using the Locality-constrained Linear Coding formula [27]:

$$\mathbf{w}_i' = \left(\mathbf{X}_i^{(i)T}\mathbf{X}_i^{(i)} + \sigma\,\mathbf{P}^{(i)}\right)^{-1}\mathbf{X}_i^{(i)T}\mathbf{x}_i, \tag{4.8}$$

where $\mathbf{P}^{(i)}$ is a diagonal matrix and $P_{jj}^{(i)}$ is given by:

$$P_{jj}^{(i)} = \frac{1}{W_{ij}}. \tag{4.9}$$

Thus, the procedure of TPSRLS is shown in Algorithm 2.

4.4 Kernelized variants

In this section, we introduce two kernelized versions of the proposed SRLS method.

4.4.1 Hilbert space

The motivation behind the use of Kernel trick is that the original SRLS assumes a linear model for the data self-representativeness. However, for some data sets this assumption may not be very realistic. Thus, by adopting a non-linear model for the data self-representativeness, it is expected that the resulting coding coefficients can better quantify the similarity among samples and hence, better graphs can be obtained whenever data have non-linear distribution.

Input: The original data matrix $\mathbf{X}$, a given rough weight matrix $\mathbf{W}_0$, λ, and ρ
Output: TPSRLS graph $\mathbf{W}'$

First Phase: SRLS graph;
Calculate SRLS graph $\mathbf{W}$ using Algorithm 1;

Second Phase: TPSRLS graph;
for $i = 1, \ldots, n$ **do**
  Compute a sample based threshold using Eq.(4.7);
  Form the set of samples $R_i = \{\mathbf{x}_j \mid W_{ij} > Th(\mathbf{w}_i)\}$;
  Form the reduced $d \times n_s$ data matrix $\mathbf{X}_s^{(i)}$ from R_i where $|R_i| = n_s$;
  Form the new diagonal weight matrix $\mathbf{P}^{(i)}$ using Eq.(4.9) ;
  Calculate the vector $\mathbf{w}'_i$ as $\mathbf{w}'_i = \left(\mathbf{X}_s^{(i)T}\mathbf{X}_s^{(i)} + \sigma\mathbf{P}^{(i)}\right)^{-1} \mathbf{X}_s^{(i)T}\mathbf{x}_i$;
  Set the i-th row of $\mathbf{W}'$ by using $\mathbf{w}'_i$;
end

Algorithm 2: TPSRLS graph construction

Let $\Phi : \mathbf{X} \rightarrow \Phi(\mathbf{X})$ be a non-linear projection that maps original samples onto a space of high dimension. In the new space, the data are represented by the matrix $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_n)]$. Let $K_{ij} = \phi^T(\mathbf{x}_i)\,\phi(\mathbf{x}_j)$ be a similarity measure between samples $\mathbf{x}_i$ and $\mathbf{x}_j$. $K(\cdot,\cdot)$ can be Gaussian, polynomial, or any other function that satisfy Mecer's conditions. It is easy to show that the matrix $\mathbf{K}$ will be given by $\Phi^T\,\Phi$.

The kernelized version of the SRLS method can be obtained by replacing the data with their non-linear projection. Thus, we have:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_\mathbf{B}\,\mathbf{B}^T). \tag{4.10}$$

After some algebraic manipulation, we can get a similar expression for the criterion to be minimized. Indeed, the term $\mathbf{X}^T\,\mathbf{X}$ is replaced by $\mathbf{K} = \Phi^T\,\Phi$. Thus, we can use a similar iterative algorithm. The steps of the kernel version are described in Algorithm 3.

4.4.2 Column generation

Column generation replaces each sample $\mathbf{x}_i$ by a vector of similarities of that sample with the training samples. The data matrix $\mathbf{X}$ is thus replaced by the matrix $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n]$, where each $\mathbf{g}_i$ is formed by $G_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j), j = 1, ..., n$ is one kind of similarity between the sample $\mathbf{x}_i$ with sample $\mathbf{x}_j$.

The optimization problem becomes:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\mathbf{G} - \mathbf{G}\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_\mathbf{B}\,\mathbf{B}^T). \tag{4.11}$$

The solution to Eq.(4.11) is again similar to Algorithm 1, replacing $\mathbf{X}$ by $\mathbf{G}$. We can notice that $\mathbf{G} = \mathbf{K}$.

Input: The original data matrix $\mathbf{X}$, a given rough weight matrix $\mathbf{W}_0$, λ, and ρ
Output: Kernel SRLS graph $\mathbf{W}$

Set $t = 0$, $\mathbf{B}_0 = \mathbf{W}_0$, ϵ (small positive threshold);
Compute $\mathbf{K}$, where $K_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j)$;
repeat
  Calculate the graph Laplacian of $\mathbf{W}_t$, i.e. $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$;
  Get $\mathbf{B}_{t+1}$ by solving the Sylvester equation $(\mathbf{K} + \lambda\,\mathbf{I})\mathbf{B} + \rho\,\mathbf{B}\,\mathbf{L}_t = \mathbf{K}$;
  Calculate new $\mathbf{W}_{t+1}$ from $\mathbf{B}_{t+1}$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2 / n^2 < \epsilon$;
Get final Kernel SRLS graph $\mathbf{W} = \mathbf{W}_t$.

Algorithm 3: Kernel SRLS graph construction

The steps of the column generation algorithm are described in Algorithm 4.

Input: The original data matrix $\mathbf{X}$, a given rough weight matrix $\mathbf{W}_0$, λ, and ρ
Output: CG SRLS graph $\mathbf{W}$

Set $t = 0$, $\mathbf{B}_0 = \mathbf{W}_0$, ϵ (small positive threshold);
Compute $\mathbf{G}$, where $G_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j)$;
repeat
  Calculate the graph Laplacian of $\mathbf{W}_t$, i.e. $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$;
  Get $\mathbf{B}_{t+1}$ by solving the Sylvester equation
  $(\mathbf{G}^T\mathbf{G} + \lambda\,\mathbf{I})\mathbf{B} + \rho\,\mathbf{B}\,\mathbf{L}_t = \mathbf{G}^T\mathbf{G}$;
  Calculate new $\mathbf{W}_{t+1}$ from $\mathbf{B}_{t+1}$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2 / n^2 < \epsilon$;
Get final CG SRLS graph $\mathbf{W} = \mathbf{W}_t$.

Algorithm 4: Column Generation of SRLS graph construction

4.5 A direct solution to Kernel SRLS

In the previous section, we solve the Kernel SRLS problem in a cascade way. In this section, we try to solve it in one step. Let's look into the Kernel SRLS criterion: $\min_{\mathbf{B}} \{\|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_{\mathbf{B}}\,\mathbf{B}^T)\}$.

Since $\mathbf{B}$ is asymmetric, its Laplacian matrix will be given by: $\mathbf{L}_{\mathbf{B}} = \mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T$. Then the above criterion becomes:

$$\begin{aligned} &\min \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 \\ &\quad + \rho\,\mathrm{Tr}(\mathbf{B}(\mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T)\mathbf{B}^T), \\ &s.t.\ B_{ij} \geq 0 \,\text{and}\, B_{ii} = 0, \end{aligned} \tag{4.12}$$

where $\mathbf{D}_r$ and $\mathbf{D}_c$ are two diagonal matrix whose elements are the row sums and the column sums of $\mathbf{B}$, respectively.

Note that minimizing Eq.(4.12) is subject to $B_{ij} \geq 0$. Let $\mathbf{Z}$ with $Z_{ij} \geq 0$ be the corresponding Lagrange multipliers. Consider the Lagrange function $F(\mathbf{B})$ as:

$$\begin{aligned} F(\mathbf{B}) =& \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_F^2 \\ &+ \rho\,\mathrm{Tr}(\mathbf{B}(\mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T)\mathbf{B}^T) + \mathrm{Tr}(\mathbf{Z}\,\mathbf{B}^T). \end{aligned} \tag{4.13}$$

It's easy to show that we have $\mathrm{Tr}(\mathbf{Z}\,\mathbf{B}^T) = \sum Z_{ij} B_{ij}$.

Then taking the derivative of $F(\mathbf{B})$ with respect to $\mathbf{B}$ leads to:

$$\begin{aligned} \frac{\partial F(\mathbf{B})}{\partial \mathbf{B}} =& 2\mathbf{K}\mathbf{B} - 2\mathbf{K} + 2\lambda\mathbf{B} + \rho(\mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B})) \\ &- 2\rho(\mathbf{B}\mathbf{B} + \mathbf{B}^T\mathbf{B} + \mathbf{B}\mathbf{B}^T) + \mathbf{Z}, \end{aligned} \tag{4.14}$$

where $\mathbf{M}_r(\mathbf{B}) = \partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)/\partial \mathbf{B}$, $\mathbf{M}_c(\mathbf{B}) = \partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)/\partial \mathbf{B}$ and $\mathbf{K} = \Phi^T\Phi$. One suggestion for $\mathbf{K}$ is Gaussian Kernel, i.e. $K_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$.

It's difficult to have simple expressions for $\mathbf{M}_r(\mathbf{B})$ and $\mathbf{M}_c(\mathbf{B})$. But we can have a detailed look into every elements of them. Let $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$. Then we can have $\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T) = \sum_k D_{r(k)}\|\mathbf{b}_k\|^2$, where $D_{r(k)}$ is the k-th diagonal elements of $\mathbf{D}_r$, i.e. $D_{r(k)} = \sum_t B_{kt}$. Similarly, $\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T) = \sum_k D_{c(k)}\|\mathbf{b}_k\|^2$, where $D_{c(k)} = \sum_t B_{tk}$.

Then the derivative of $\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)$ and $\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)$ with respect to B_{ij} can be obtained:

$$\frac{\partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)}{\partial B_{ij}} = \|\mathbf{b}_i\|^2 + 2B_{ij}D_{r(j)}, \tag{4.15}$$

and

$$\frac{\partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)}{\partial B_{ij}} = \|\mathbf{b}_j\|^2 + 2B_{ij}D_{c(j)}. \tag{4.16}$$

With these, every element of $\mathbf{M} = \mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B})$ can be obtained:

$$\begin{aligned} M_{ij} &= (\mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B}))_{ij} \\ &= \frac{\partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)}{\partial B_{ij}} + \frac{\partial \mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)}{\partial B_{ij}} \\ &= \|\mathbf{b}_i\|^2 + \|\mathbf{b}_j\|^2 + 2B_{ij}D_{r(j)} + 2B_{ij}D_{c(j)}. \end{aligned} \tag{4.17}$$

Then $\partial F(\mathbf{B})/\partial Bij$ can be:

$$\begin{aligned}\frac{\partial F(\mathbf{B})}{\partial B_{ij}} =&(2\mathbf{KB} - 2\mathbf{K} + 2\lambda\mathbf{B} - 2\rho(\mathbf{BB} + \mathbf{B}^T\mathbf{B} + \mathbf{BB}^T))_{ij} \\ &+ Z_{ij} + \rho M_{ij}.\end{aligned} \tag{4.18}$$

With $\partial F(\mathbf{B})/\partial \mathbf{B} = 0$ and the *Karush-Kuhn-Tucker* (KKT) [38, 28] condition $Z_{ij}B_{ij} = 0$, we will have:

$$(2\mathbf{KB} - 2\mathbf{K} + 2\lambda\mathbf{B} - 2\rho(\mathbf{BB} + \mathbf{B}^T\mathbf{B} + \mathbf{BB}^T) + \rho\mathbf{M})_{ij}B_{ij} = 0. \tag{4.19}$$

An iteratively updating rule on $\mathbf{B}$ is designed as

$$B_{ij} \leftarrow \frac{2K_{ij} + 2\rho(\mathbf{BB} + \mathbf{B}^T\mathbf{B} + \mathbf{BB}^T)_{ij}}{(2\mathbf{KB} + 2\lambda\mathbf{B} + \rho\mathbf{M})_{ij}} B_{ij}. \tag{4.20}$$

Thus, we can use an iterative algorithm. Note the initial $\mathbf{B}_0$ will be important since when B_{ij} will maintain 0 when it once become 0 in the iteration. One simple choice of the initial $\mathbf{B}_0$ is a matrix with all the elements is 1 except the diagonal elements is 0. In other word, $\mathbf{B}_0 = \mathbf{E} - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

The steps of the proposed method are described in Algorithm 5.

Input: The original data matrix $\mathbf{X}$, initial matrix $\mathbf{B}_0$ λ, ρ and ϵ (a small positive threshold)
Output: DSRLS graph $\mathbf{W}$

Set $t = 0$;
Calculate matrix $\mathbf{K} = \mathbf{X}^T\mathbf{X}$;
repeat
  Calculate matrix $\mathbf{M}_t$ using Eq.(4.17);
  Update matrix $\mathbf{B}$ with updating rule
  $B_{ij} \leftarrow \frac{2K_{ij}+2\rho(\mathbf{B}_t\mathbf{B}_t+\mathbf{B}_t^T\mathbf{B}_t+\mathbf{B}_t\mathbf{B}_t^T)_{ij}}{(2\mathbf{KB}_t+2\lambda\mathbf{B}+\rho\mathbf{M}_t)_{ij}} B_{ij}$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2/n^2 < \epsilon$;
Calculate final DSRLS graph matrix $\mathbf{W} = (|\mathbf{B}| + |\mathbf{B}^T|)/2$;

Algorithm 5: DSRLS graph construction

4.6 Performance evaluation

In general, the estimated graph alone cannot lend itself to an easy assessment of the method that constructs it. Indeed, given a real data set as well as a machine learning task that uses this data set, it is very often challenging if not unfeasible to know in advance the ideal graph for that data set and for

that task. Thus, in our work, the graph-construction methods are assessed by the performance of the post-graph construction tasks. In this section, we evaluate the proposed graph construction methods in two main applications of computer vision: face recognition and object categorization. For both problems, we use semi-supervised label propagation based on the built graph. The performance will be mainly given by recognition and classification accuracy. The construction methods used for comparison are kNN graph [5], LLE graph [88], ℓ_2 graph [105], Standard ℓ_1 graph (ℓ_1-s) [65], Robust ℓ_1 graph (ℓ_1-r) [94], and WRLS graph [26].

In this section, we evaluate the proposed algorithm on several real databases. To this end, four face databases are used: ORL, FERET, Extended Yale B and PF01, and one object database COIL-20 is used.

4.6.1 Comparison among several graph construction methods

To evaluate the performance of our proposed methods, we compare them with several other graph construction methods including KNN graph and LLE graph, standard ℓ_1 graph and robust ℓ_1 graph, the simple linear coding graph (ℓ_2 graph) and WRLS method. Our proposed methods are: SRLS, TPSRLS, SRLS-CG, SRLS-K, and DSRLS. For every graph construction method, several values for the parameter are used. We then report the best recognition accuracy (best average recognition rate) of all methods from the best parameter configuration.

kNN and LLE methods have the neighborhood size parameter k. WRLS method has the parameter λ. The five proposed methods have two parameters λ and ρ, the initial $\mathbf{W}_0$ and the threshold ϵ to set. In our experiments, k is chosen from 3 to 60 with a step of 3, λ in the WRLS method is simply set to 1, λ and ρ are chosen from $\{10^{-4}, 10^{-3}, 0.01, 0.1, 1, 10\}$, $\mathbf{W}_0$ is set by the kNN graph when $k = 3$ and $\epsilon = 10^{-6}$. For the two kernel versions of SRLS, we use the Gaussian Kernel $sim(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$. The parameter of the Gaussian is set to the average distances among the samples.

After we construct the graphs on the original data, we run the label propagation methods LGC and GRF as classifiers. In LPDGL, the parameter β and γ are simply set to 1. For label propagation, we randomly split the whole data set into a labeled part and unlabeled part and repeat this process 10 times. The final performance (recognition rate) is given by the average.

Tables 4.1, 4.2, 4.3, 4.4, 4.5 illustrate the method comparison obtained with the GRF label propagation method on five datasets.

Table 4.1 shows the mean recognition rates over the 10 random splits on ORL with different graph construction methods. In this table, different label numbers l are used.

For ORL data set, the column generation version of SRLS performs best among all the graph construction methods. For this dataset, the two phase method and kernelized variants have improved the quality of the final graph. The direct method DSRLS has also improved its quality.

Table 4.1: Recognition rates (%) on ORL by GRF.

Method\l	1	2	3	4	5
kNN [5]	75.1	85.2	89.6	92.3	94.2
LLE [88]	74.1	83.7	88.6	93.1	96.2
ℓ_1-s [65]	76.8	86.7	91.4	93.9	95.6
ℓ_1-r [94]	64.8	80.4	87.6	91.2	93.8
ℓ_2 [105]	66.3	80.9	86.9	91.0	93.5
WRLS [26]	60.3	76.8	85.6	90.5	93.1
SRLS	67.4	82.5	87.9	91.0	93.4
TPSRLS	74.6	85.5	91.1	93.7	95.5
SRLS-CG	**83.8**	**89.6**	**93.7**	**95.3**	**97.1**
SRLS-K	71.1	84.1	89.3	92.8	94.3
DSRLS	75.1	85.9	92.0	94.2	96.3

Table 4.2 shows the mean recognition rates on FERET with different graph construction methods. In the table, the number of labeled samples varies from 1 to 5 samples per class. On FERET, the kernel version of SRLS method performs best among all the graph construction methods. We can observe that the performance of five proposed methods was similar to or better than the best sparse method (i.e., ℓ_1-s).

Table 4.3 shows the mean recognition rates on Extended Yale B with different graph construction methods. Several label percentages are used in the interval 10%-30%. For Extended Yale B data set, the robust ℓ_1 graph performs best among all the graph construction methods. SRLS method performs a slightly less than the robust ℓ_1 graph. The recognition rates of the TPSRLS is quite close to that of the robust ℓ_1 graph.

Table 4.4 shows the mean recognition rates on PF01. Several label numbers are used (2-10 samples). On PF01, the recognition rates of SRLS is quite close to that of the robust ℓ_1 graph. The column generation one and the kernel one outperform the other graph construction methods.

Table 4.5 shows the mean recognition rates on COIL-20 with different graph construction methods. 6, 9 and 12 labeled samples per class are used for label propagation. On COIL-20, TPSRLS performs best among all the methods in general.

Table 4.6 shows the mean recognition rates on ORL with different graph construction methods when the label propagation method is the LGC. On ORL, the basic kernel version of SRLS performs best among all the graph construction methods.

Table 4.7 shows the mean recognition rates on FERET with different graph construction methods when the label propagation method is the LGC. On FERET, the column generation version of SRLS method performs best among all the graph construction methods.

Table 4.2: Recognition rates (%) on FERET by GRF.

Method\l	1	2	3	4	5
kNN	22.4	30.9	39.0	45.2	51.0
LLE	31.3	43.9	55.7	66.9	72.7
ℓ_1-s	40.2	56.0	66.5	72.7	74.8
ℓ_1-r	39.1	57.3	70.6	79.3	81.4
ℓ_2	34.7	52.5	64.3	69.9	71.5
WRLS	39.1	57.0	71.4	78.6	80.4
SRLS	37.9	54.8	66.6	71.9	72.5
TPSRLS	36.5	55.4	69.7	78.5	80.6
SRLS-CG	47.4	65.7	77.7	84.3	85.3
SRLS-K	**47.5**	**65.8**	**78.0**	**84.7**	**86.4**
DSRLS	44.5	61.4	72.1	79.4	81.9

Table 4.3: Recognition rates (%) on Extended Yale B by GRF.

Method\l	6	13	19
kNN	81.7	89.1	84.1
LLE	66.3	74.9	89.3
ℓ_1-s	79.3	86.6	90.0
ℓ_1-r	**90.5**	**94.0**	**96.1**
ℓ_2	85.4	93.3	95.3
WRLS	82.7	91.3	94.7
SRLS	86.1	93.4	95.5
TPSRLS	88.5	93.4	95.7
SRLS-CG	83.7	91.8	95.0
SRLS-K	82.8	91.5	95.4
DSRLS	87.2	93.9	91.7

Table 4.4: Recognition rates (%) on PF01 by GRF.

Method\l	2	4	6	8	10
kNN	35.5	43.3	47.0	50.6	53.8
LLE	34.6	43.6	51.2	64.9	73.7
ℓ_1-s	40.3	50.4	55.9	59.3	63.6
ℓ_1-r	57.5	69.7	75.6	80.1	86.0
ℓ_2	54.3	69.8	76.6	80.7	85.1
WRLS	54.7	69.6	75.5	79.7	85.1
SRLS	57.2	71.6	77.4	81.2	85.8
TPSRLS	59.0	71.1	78.0	82.4	87.7
SRLS-CG	**60.5**	**74.4**	**79.6**	83.6	88.4
SRLS-K	58.8	73.9	79.3	**83.7**	**89.0**
DSRLS	47.9	62.3	68.1	72.7	78.7

Table 4.5: Recognition rates (%) on COIL-20 by GRF.

Method\l	6	9	12
kNN	87.8	88.9	90.3
LLE	91.6	94.3	96.7
ℓ_1-s	57.0	59.1	60.4
ℓ_1-r	92.9	94.9	**96.9**
ℓ_2	84.9	87.7	90.2
WRLS	83.7	85.3	88.6
SRLS	85.5	87.9	90.5
TPSRLS	**93.7**	**95.3**	96.7
SRLS-CG	90.9	92.3	94.1
SRLS-K	90.2	92.1	94.0

Table 4.8 shows the mean recognition rates on Extended Yale B with different graph construction methods when the label propagation method is the LGC. On Extended Yale B, the column generation version of SRLS method performs best among all the graph construction methods.

Table 4.6: Recognition rates (%) on ORL by LGC.

Method\l	1	2	3	4	5
kNN	67.0	81.5	85.2	90.4	92.4
LLE	69.0	83.1	85.0	86.2	89.0
ℓ_1-s	68.6	83.0	89.4	92.5	94.6
ℓ_1-r	47.6	65.6	77.3	84.2	88.8
ℓ_2	68.8	81.4	86.7	90.5	93.1
WRLS	53.8	71.6	80.4	87.0	90.4
SRLS	70.0	82.7	88.1	91.0	93.5
TPSRLS	69.7	82.5	88.9	92.3	94.0
SRLS-CG	70.7	83.7	89.2	92.4	94.3
SRLS-K	**75.1**	**86.1**	**90.2**	**93.7**	**95.9**

Table 4.9 shows the mean recognition rates on Extended Yale B with different graph construction methods when the label propagation method is the LPDGL. Generally, the proposed algorithms can be better than the other graph construction method on Extended Yale B by LPDGL.

Table 4.10 shows the mean recognition rates on Extended Yale B with 6 labeled samples when different label propagation methods are used. In general, the proposed methods can outperform the other graph construction methods whatever label propagation method is used.

Analysis of results: According to the results reported in the previous tables, we can make the following observations:

Table 4.7: Recognition rates (%) on FERET by LGC.

Method\l	1	2	3	4	5
kNN	19.6	27.9	34.7	40.8	45.8
LLE	32.3	49.5	62.4	71.8	75.8
ℓ_1-s	45.7	61.4	71.2	77.0	78.7
ℓ_1-r	30.2	47.8	62.0	73.7	77.7
ℓ_2	44.9	59.5	68.7	72.6	72.7
WRLS	32.2	50.3	63.5	71.3	73.9
SRLS	45.2	59.5	69.0	73.4	73.6
TPSRLS	37.1	54.4	67.5	76.6	79.4
SRLS-CG	**50.3**	**67.7**	**78.6**	**85.2**	**87.0**
SRLS-K	50.0	67.0	78.4	84.7	86.0

Table 4.8: Recognition rates (%) on Extended Yale B by LGC.

Method\l *(%)*	10	20	30
kNN	81.2	83.3	84.9
LLE	84.1	88.1	89.7
ℓ_1-s	83.2	89.3	91.9
ℓ_1-r	75.9	86.3	91.8
ℓ_2	88.1	92.3	94.4
WRLS	82.7	91.4	94.6
SRLS	**89.2**	**94.1**	**95.8**
TPSRLS	87.2	92.9	95.2
SRLS-CG	84.5	92.2	92.5
SRLS-K	85.7	92.4	95.2

Table 4.9: Recognition rates (%) on Extended Yale B by LPDGL.

Method\l *(%)*	10	20	30
kNN	81.3	83.7	85.1
LLE	84.5	86.2	87.8
ℓ_1-s	84.8	91.5	94.9
ℓ_1-r	83.2	89.1	91.7
ℓ_2	89.3	93.8	95.2
WRLS	84.6	91.4	94.5
SRLS	89.3	**93.9**	95.4
TPSRLS	**89.8**	93.6	**95.5**

Table 4.10: Recognition rates (%) on Extended Yale B with 6 labeled by different label propagation algorithms.

Method	GRF	LGC	LPDGL
kNN	81.7	81.2	81.3
LLE	66.3	84.1	84.5
ℓ_1-s	79.3	83.2	84.8
ℓ_1-r	**90.5**	75.9	83.2
ℓ_2	85.4	88.1	89.3
WRLS	82.7	82.7	84.6
SRLS	86.1	**89.2**	89.3
TPSRLS	88.5	87.2	**89.8**

- In general, the proposed methods (including the two kernelized versions) outperform the other methods. The two phase SRLS can improve the recognition rate for both label propagation methods (GRF and LGC).
- The proposed SRLS is almost always superior to the classic ℓ_2 method suggesting that Laplacian smoothness has contributed to get more informative graphs.
- In most cases, the kernelized versions of SRLS provided the best performances except for Extended Yale B.
- The proposed SRLS graph can be competitive to the robust ℓ_1 graph. It can outperform it when LGC is used. We stress the fact that it is more efficient to construct SRLS graph than robust ℓ_1 graph.
- In general, DSRLS performance depends on the datasets used. in general, it outperforms the SRLS and many existing state-of-the art methods.
- For the used datasets, the kernelized versions of the SRLS gave the best results in most cases.

4.6.2 Stability of the proposed method

In this section, we will empirically show that independently of the initial weight matrix $\mathbf{W}_0$ used by the Laplacian smoothness criterion, the successive minimizations adopted by the SRLS method will provide the same graph. This will validate our assumption for the adopted algorithm for SRLS, TPSRLS, and the two kernel versions. To this end, we conduct the following experiment. We initialize the SRLS method with different weight matrices $\mathbf{W}_0$. Here we use the kNN graph, the robust ℓ_1 graph and a random one for initial graphs. For each such initialization, we run the iterative SRLS method in order to compute the optimal matrix $\mathbf{B}$. We then compare the obtained graphs as well as performance obtained with them. To evaluate the trend of Laplacian smoothness term, we calculate the values of $\mathrm{Tr}(\mathbf{BLB}^T)$ in every iteration with a fixed $\mathbf{L}$. The fixed Laplacian matrix is set for two types of graphs (KNN

graph, robust l1 graph). We also use the graph that is iteratively provided by SRLS.

In this comparison, we use the performance of the GRF label propagation method based on the affinity matrices $\mathbf{B}$ obtained at convergence (for all types of initialization). We also calculate the differences between the $\mathbf{B}$'s for every iteration and the differences between the optimal $\mathbf{B}$ computed with different initial weight matrices. For PF01, λ is set to 0.01 and $\rho = 0.1$, while for the other data sets, we set $\lambda = 1$, $\rho = 1$. We used 3 labeled samples per class with LGC for FERET and 4 labeled samples with GRF for PF01.

Table 4.11-Table 4.13 show the obtained results on FERET and Table 4.14-Table 4.16 show the results on PF01. Table 4.17 illustrates the values of $\mathrm{Tr}(\mathbf{BLB}^T)$ in every iteration for five different data sets.

Table 4.11 and Table 4.14 show the average recognition rates associated with the affinity matrix $\mathbf{B}$ obtained at each iteration of the SRLS method. In each Table, three different initial weight matrices are used.

Table 4.12 and Table 4.15 illustrate the difference between the graphs obtained in two consecutive iterations $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$, that is the Frobenius norm $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|^2$, $\mathbf{B}_0 = \mathbf{W}_0$. Table 4.13 and Table 4.16 show the difference between the optimal $\mathbf{B}$s obtained by different initial weight matrices $\mathbf{W}_0$. Here, $\mathbf{B}^*_{knn}$ refer to the optimal $\mathbf{B}$ obtained by SRLS when the initial weight matrix was the kNN. $\mathbf{B}^*_{\ell_1}$ refer to the optimal $\mathbf{B}$ obtained by SRLS when the initial weight matrix was set to the robust ℓ_1 graph. $\mathbf{B}^*_{rand}$ refers to the optimal $\mathbf{B}$ obtained by SRLS when the initial weight matrix was set to completely random graph.

Table 4.11: Recognition rates (%) of each $\mathbf{B}_i$ in iteration on FERET with 3 labeled every class by LGC.

$\mathbf{W}_0 \backslash$Iteration (t)	0	1	2	3	4
kNN	34.7	67.0	69.0	69.0	69.0
ℓ_1-r	62.0	69.4	69.0	69.0	69.0
random	0.6	68.6	69.0	69.0	69.0

Table 4.12: Differences between $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$ in iteration on FERET.

$\mathbf{W}_0$	ϵ_1	ϵ_2	ϵ_3	ϵ_4
kNN	1.8e4	6.2e-1	2.8e-4	7.1e-7
ℓ_1-r	3.0e2	2.3e-1	7.8e-5	2.1e-7
random	5.7e5	1.2e-3	2.4e-5	7.8e-8

Table 4.13: Differences between optimal **B**s on FERET.

$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{\ell_1}\|^2$	$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{rand}\|^2$
2.6e-9	1.5e-9

Table 4.14: Recognition rates (%) of $\mathbf{B}_t$ in iteration on PF01 with 4 labeled every class by GRF.

$\mathbf{W}_0$\Iteration (t)	0	1	2	3	4	5
kNN	37.9	70.6	71.4	71.4	71.4	71.4
ℓ_1-r	69.6	72.2	71.4	71.4	71.4	71.4
random	0.9	71.0	71.4	71.4	71.4	-

Table 4.15: Differences between $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$ in iteration on PF01.

$\mathbf{W}_0$	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
kNN	2.2e4	1.9	7.1e-4	1.8e-6	5.9e-9
ℓ_1-r	5.3e2	1.4	4.5e-4	1.0e-6	3.2e-9
random	9.6e5	4.3e-2	1.1e-4	3.3e-7	-

Table 4.16: Differences between optimal **B**s on PF01.

$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{\ell_1}\|^2$	$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{rand}\|^2$
2.8e-11	1.1e-9

According to the recognition rates in Table 4.11 and Table 4.14, we can see that in less than 5 iterations, the optimal recognition rates will be obtained. As shown in Table 4.12 and Table 4.15, the value of $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|^2$ will be less than 1e-3 around 3 or 4 iterations which means the proposed method can obtain the optimal **B** within 3 or 4 iterations. The results in Table 4.13 and Table 4.16 show that the values of the difference between the optimal **B**s obtained by different initial **W**s is quite small which means that the same optimal **B** can be obtained regardless of the initial $\mathbf{W}_0$. From the results depicted in the above tables, we can conclude that regardless of the initialization used, the SRLS is able to estimate the same optimal graph. In particular, Table 4.13 and Table 4.16 show that different initial $\mathbf{W}_0$ will lead to the same optimal SRLS graph **B**. This fact makes the SRLS not dependent on a given initialization. Thus, SRLS can provide the same optimal graph regardless of the initial affinity matrix used as a Laplacian. This can be explained by the fact that data self-representativeness is a strong constraint.

Table 4.17: Smoothness value, $Tr(\mathbf{B}_t \mathbf{L} \mathbf{B}_t^T)$, at each iteration for different Laplacians.

Dataset	$\mathbf{L}$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
ORL	kNN	5915.1	10.9	9.9	9.9	9.9
	ℓ_1-r	5899.8	13.5	10.8	10.8	10.8
	$\mathbf{L}_t$	5915.1	24.2	19.4	19.3	19.3
FERET	kNN	30106.1	36.0	33.2	33.2	-
	ℓ_1-r	31970.3	46.5	39.5	39.5	-
	$\mathbf{L}_t$	30106.1	66.6	56.5	56.5	-
Extended Yale B	kNN	30909.1	27.2	22.4	22.4	22.4
	ℓ_1-r	31325.5	34.5	26.6	26.7	26.7
	$\mathbf{L}_t$	30909.1	74.3	57.6	57.5	57.5
PF01	kNN	33134.4	159.1	154.9	155.0	155.0
	ℓ_1-r	35482.6	162.0	148.5	148.6	148.6
	$\mathbf{L}_t$	33134.4	244.8	221.2	221.4	221.4
COIL-20	kNN	4541.7	8.6	7.7	7.8	7.8
	ℓ_1-r	4567.9	11.4	9.3	9.3	9.3
	$\mathbf{L}_t$	4541.7	21.3	17.1	17.1	17.1

As illustrated in Table 4.12 and Table 4.15, the proposed methods converge within 5 iterations. Actually, the SRLS algorithm can almost obtain the optimal $\mathbf{B}$ in 2 iterations.

Finally, Table 4.17 illustrates the evolution of the smoothness term $\mathrm{Tr}(\mathbf{B}_t\mathbf{L}\mathbf{B}_t^T)$ when the graph asymmetric matrix $\mathbf{B}$ is estimated by the recursive minimization of Eq.(4.3). The results correspond to five datasets and to three graph Laplacians: kNN graph, robust ℓ_1 graph and SRLS graph $\mathbf{B}_t$ in every iteration. For kNN graph and robust ℓ_1 graph, the corresponding Laplacian matrix $\mathbf{L}$ is kept fixed. From this table, we can see that, independently of the used Laplacian, the smoothness value $\mathrm{Tr}(\mathbf{B}\mathbf{L}\mathbf{B}^T)$ decreases by the successive minimizations, which indicates that the Laplacian smoothness term of the estimated graph is really decreasing over the recursions.

4.6.3 Sensitivity to parameters

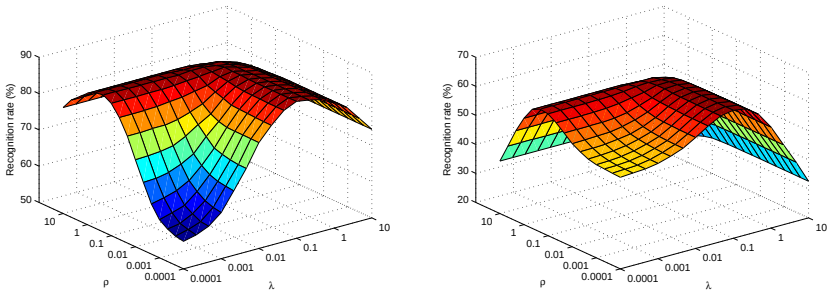
In the SRLS algorithm, parameters λ and ρ should be set. We aim to study the recognition rates obtained by SRLS when these two parameters vary. Figure 4.2 illustrates the recognition rates with different parameter value for ORL, FERET and Extended Yale B.

On ORL, 3 samples for each person are labeled. According to Fig.4.2a, λ and ρ near 1 will lead to a better graph for final recognition rates.

On FERET, 3 samples for each person are labeled. According to Fig.4.2b, λ and ρ near 1 will lead to a better graph for final recognition rates.

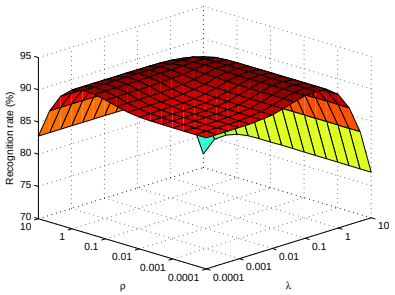
On Extended Yale B, 13 samples for each person are labeled. According to Fig.4.2c, λ and ρ near 1 will lead to a better graph for final recognition rates and the differences are small when λ and ρ are not larger than 1.

We can observe that the optimal domain for the two parameters λ and ρ is almost the same for the three face datasets. This tends to confirm that using SRLS and its variants does not need a tedious task for choosing the two parameters.



(a) ORL

(b) FERET



(c) Extended Yale B

Fig. 4.2: Recognition rates variation different values of parameter λ and ρ on ORL, FERET and Extended Yale B.

4.6.4 Computational complexity and CPU time

In the SRLS algorithm, the complexity of computing $\mathbf{X}^T\mathbf{X}$ before iteration will be $O(d\,n^2)$. In each iteration, SRLS mainly solves the Sylvester equation. According to [42, 69], the computational complexity to solve the Sylvester equation is $O(n^3)$. Thus, the computational complexity of SRLS will be $O(d\,n^2 + \tau n^3)$, where τ is the number of iterations. In this case, when n is

large, i.e. the number of the train data is huge, the graph construction methods will be time consuming. To solve the problem of huge data, data sample can be compacted using instance selection or data representation methods such as described in article [25].

Table 4.18 illustrates the CPU time needed by the SRLS method and the robust ℓ_1 graph method for constructing the graph associated with ORL, FERET, PF01 and Extended Yale B data sets. We set $\lambda = \rho = 1$ in SRLS. All tests are conducted using Matlab codes running on a PC equipped with an Intel Core i7-4771 CPU at 3.5GHz and 8 GB of RAM. According to Table 4.18, the proposed SRLS methods is far more efficient than the robust ℓ_1 graph method.

Table 4.18: CPU time (s).

Dataset\Method	SRLS	ℓ_1-r
ORL	3.0	50.8
FERET	84.4	1689.2
PF01	167.0	2421.4
Extended Yale B	209.9	1579.4

4.7 Conclusion

In this chapter, we proposed a graph construction method based on self representation and Laplacian smoothness. The proposed method preserves the smoothness of the self representation matrix. Moreover we introduce the column generation and kernel version of SRLS. According to the experimental results, the proposed methods can handle real data sets with a better performance than other graph construction methods in general. The proposed methods is more efficient but can perform as well as and even better than the robust ℓ_1 graph.

One of the disadvantages of the proposed method is the choice of parameters. Although the experiments show that the parameters λ and ρ in the proposed criterion should be near 1, still no evidences show that the parameters can be decided directly.

5

Sparse graph with Laplacian Smoothness

Abstract

In this chapter, we introduce a new sparse graph construction method using Laplacian smoothness as a constraint (SGLS). We also propose a kernelized version of the SGLS method and a direct solution to SGLS.

Contents

5.1 Sparse graph with Laplacian Smoothness

Adding constraints to the ℓ_1 graph construction can lead to better performances as it was shown in [19]. In the latter work, the method needs the setting of a similarity matrix that will be used to constrain the unknown sparse affinity matrix. In practice, knowing this optimal matrix is very challenging. In this chapter, we propose a constrained sparse graph that uses a natural constraint given by the Laplacian smoothness. In the construction of classic sparse graph, the goal is to obtain a representation matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$, which is the solution to the following ℓ_1 problem:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \{\|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda \|\mathbf{B}\|_1\}. \tag{5.1}$$

$\mathbf{B}$ can be regarded as the asymmetric weight matrix associated with the training samples. In other words, $\mathbf{B}$ is the coding matrix obtained when the dictionary is set to the data themselves. For every sample $\mathbf{x}_i$, $\mathbf{b}_i$ is the sparse

coding vector needed to construct the sample $\mathbf{x}_i$, from the whole data set of samples.

In ideal cases, the coding vectors of similar original samples should also be similar. For instance, if $\mathbf{x}_i$ is very similar to $\mathbf{x}_j$, then the corresponding coding vectors $\mathbf{b}_i$ and $\mathbf{b}_j$ should be close to each other. Mathematically, this property is known by Laplacian smoothness criterion. For all pairs, this criterion is given by:

$$\sum_{i,j} \|\mathbf{b}_i - \mathbf{b}_j\|^2 B_{ij} = \mathrm{Tr}(\mathbf{B}\,(\mathbf{D} - \mathbf{W})\,\mathbf{B}^T) = \mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T), \tag{5.2}$$

where $\mathbf{L}$ is the graph Laplacian of the weight matrix $\mathbf{W}$, i.e. $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$ and $\mathbf{W}$ is a given graph.

We propose to construct an affinity matrix $\mathbf{B}$ that simultaneously uses data sparse representation and Laplacian smoothness. Thus, our coding matrix will be the solution to the following minimization problem:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \left\{ \frac{1}{2}\|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_{\mathbf{B}}\,\mathbf{B}^T) \right\}, \tag{5.3}$$

where $\mathbf{L}_{\mathbf{B}}$ is the Laplacian matrix of the affinity matrix $\mathbf{B}$, and λ and ρ are two positive balance parameters.

Our proposed method (SGLS) is to solve the problem depicted in Eq.(5.3). Notice that if $\mathbf{L}_{\mathbf{B}}$ is fixed, the functional in the Eq.(5.3) will become an ℓ_1 problem which can be solved by the Alternating Direction Method of Multipliers (ADMM). However, the graph Laplacian is unknown since the graph itself is unknown. To solve that, we propose a new recursive method to get the optimal coding matrix (equivalently the graph). The intuition behind our idea is as follows. If we start with a rough graph, its corresponding Laplacian will impose smoothness of the vectors of the unknown coding matrix $\mathbf{B}$.

Solving for $\mathbf{B}$ using the optimization problem in Eq.(5.3) with a fixed Laplacian matrix, we obtain a solution that satisfies data self-representation, regularized norm, and the Laplacian smoothness. The derived symmetric graph for the current estimated $\mathbf{B}$ will be better graph (in the sense of Laplacian smoothness). Therefore, successive minimizations having the form of Eq.(5.3) will improve the estimation of the coding matrix from which the final graph is derived.

Firstly, a rough weight matrix $\mathbf{W}$ is computed by using any traditional graph construction method[1]. The graph Laplacian is then calculated by $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

Secondly, the matrix $\mathbf{B}$ is estimated by minimizing the following criterion:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \left\{ \frac{1}{2}\|\mathbf{X} - \mathbf{X}\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T) \right\}. \tag{5.4}$$

[1] The experimental section will show that even random graphs can also be used as initial graphs.

We can apply Alternating Direction Method of Multipliers to solve the Eq.(5.4). To follow the general form of ADMM, we employ a variable into the objective such that:

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{X}-\mathbf{X}\,\mathbf{B}\|_F^2+\lambda\,\|\mathbf{C}\|_1+\rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T),\\ &s.t.\ \mathbf{B}=\mathbf{C}. \end{aligned} \tag{5.5}$$

To form the augmented Lagrangian of the above objective, let $Q(\mathbf{B},\mathbf{C},\mathbf{M}) = \frac{1}{2}\|\mathbf{X}-\mathbf{X}\,\mathbf{B}\|_F^2+\lambda\,\|\mathbf{C}\|_1+\rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}\,\mathbf{B}^T)+\mathrm{Tr}(\mathbf{M}^T(\mathbf{B}-\mathbf{C}))$, where $\mathbf{M}$ is the matrix of Lagrangian multipliers.

The ADMM algorithm consists of iterating the following three substeps:

1) Fix $\mathbf{C}_k$ and estimate $\mathbf{B}_{k+1}$ by vanishing the derivatives of $Q(\mathbf{B},\mathbf{C},\mathbf{M})$. This will lead to solve a Sylvester equation:

$$(\mathbf{X}^T\mathbf{X}+\gamma\,\mathbf{I})\mathbf{B}+2\rho\,\mathbf{B}\,\mathbf{L}=\mathbf{X}^T\mathbf{X}+\gamma\mathbf{C}_k-\mathbf{M}_k. \tag{5.6}$$

2) Fix $\mathbf{B}_{k+1}$ and estimate $\mathbf{C}_{k+1}$:

$$\mathbf{C}_{k+1}=\arg\min_{\mathbf{C}} Q(\mathbf{B}_{k+1},\mathbf{C},\mathbf{M}_k) \tag{5.7}$$

This is solved element wise using soft-thresholding.

3) Fix $\mathbf{B}_{k+1}$ and $\mathbf{C}_{k+1}$ and estimate $\mathbf{M}_{k+1}$:

$$\mathbf{M}_{k+1}=\mathbf{M}_k+\gamma(\mathbf{B}_{k+1}-\mathbf{C}_{k+1}) \tag{5.8}$$

The stopping condition of the ADMM can be given by clapping the number of iterations or if the current $\|\mathbf{B}-\mathbf{C}\|^2$ becomes below a predefined threshold.

Thirdly, $\mathbf{W}$ is obtained by $W_{ij}=(|B_{ij}|+|B_{ji}|)/2$.

The last two steps are repeated until $\mathbf{B}$ does not change. Thus, the procedure is summarized in Algorithm 6.

It should be noticed that by setting the parameter ρ to zero, we get a sparse graph only since Eq. (5.4) reduces to Eq.(5.1).

Several iterative schemes to solve Sylvester equations have been proposed; for methods focusing on large sparse systems one can use the method described in [12].

5.2 Kernelized variants

In this section, we introduce a kernelized version of the proposed SGLS method.

The motivation behind the use of Kernel trick is that the original SGLS assumes a linear model for sparse representation. However, for some data sets this assumption may not be very realistic. Thus, by adopting a non-linear model for sparse representation, it is expected that the resulting coding

Input: The original data matrix $\mathbf{X}$, a rough weight matrix $\mathbf{W}_0$, λ, ρ, and ϵ_1 and ϵ_2 (two small positive thresholds)
Output: SGLS graph $\mathbf{W}$

Set $t = 0$;
repeat
  Calculate the graph Laplacian of $\mathbf{W}_t$, i.e. $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$;
  Set $k = 0$, $\mathbf{C}_0$ to a random matrix, and $\mathbf{M}_0$ to $\mathbf{0}$;
  repeat *ADMM iterations*
    Get $\mathbf{B}_{t,k+1}$ by solving the following Sylvester equation
    $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{B} + 2\rho\mathbf{B}\mathbf{L}_t = \mathbf{X}^T\mathbf{X} + \gamma\mathbf{C}_k - \mathbf{M}_k$;
    Get $\mathbf{C}_{k+1} = \arg\min_{\mathbf{C}} Q(\mathbf{B}_{t,k+1}, \mathbf{C}, \mathbf{M}_k)$;
    $\mathbf{M}_{k+1} = \mathbf{M}_k + \gamma(\mathbf{B}_{t,k+1} - \mathbf{C}_{k+1})$;
    $k = k + 1$;
  until $\|\mathbf{B}_{t,k} - \mathbf{C}_k\|_F^2 < \epsilon_2$;
  $\mathbf{B}_{t+1} = \mathbf{B}_{t,k}$;
  Calculate new $\mathbf{W}_{t+1} = (|\mathbf{B}_{t+1}| + |\mathbf{B}_{t+1}^T|)/2$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2/n^2 < \epsilon_1$;
Get final SGLS graph $\mathbf{W} = \mathbf{W}_t$.

Algorithm 6: SGLS graph construction

coefficients can better quantify the similarity among samples and hence, better graphs can be obtained whenever data have non-linear distribution.

Let $\Phi : \mathbf{X} \to \Phi(\mathbf{X})$ be a non-linear projection that maps original samples onto a space of high dimension. In the new space, the data are represented by the matrix $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), ..., \phi(\mathbf{x}_n)]$. Let $K_{ij} = \phi^T(\mathbf{x}_i)\,\phi(\mathbf{x}_j)$ be a similarity measure between samples $\mathbf{x}_i$ and $\mathbf{x}_j$. $K(.,.)$ can be Gaussian, polynomial, or any other function that satisfy Mecer's conditions. It is easy to show that the matrix $\mathbf{K}$ will be given by $\Phi^T\Phi$.

The kernelized version of the SGLS method can be obtained by replacing the data with their non-linear projection. Thus, we have:

$$\mathbf{B} = \arg\min_{\mathbf{B}} \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_{\mathbf{B}}\,\mathbf{B}^T). \tag{5.9}$$

After some algebraic manipulation, we can get a similar expression for the criterion to be minimized. Indeed, the term $\mathbf{X}^T\mathbf{X}$ is replaced by $\mathbf{K} = \Phi^T\Phi$. Thus, we can use a similar iterative algorithm. The steps of the kernel version are described in Algorithm 7.

5.3 A direct solution to Kernel SGLS

In the previous section, we solve the Kernel SGLS problem in a cascade way. In this section, we try to solve it in one step. Let's look into the Kernel SGLS criterion: $\min_{\mathbf{B}} \{\|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 + \rho\,\mathrm{Tr}(\mathbf{B}\,\mathbf{L}_{\mathbf{B}}\,\mathbf{B}^T)\}$.

Input: The kernel matrix $\mathbf{K}$, a given rough weight matrix $\mathbf{W}_0$, λ, ρ, and ϵ_1 and ϵ_2 (two small positive thresholds)
Output: Kernel SGLS graph $\mathbf{W}$

Set $t = 0$;
repeat
  Calculate the graph Laplacian of $\mathbf{W}_t$, i.e. $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$;
  Set $k = 0$, $\mathbf{C}_0$ to a random matrix, and $\mathbf{M}_0$ to $\mathbf{0}$;
  repeat *ADMM iterations*
    Get $\mathbf{B}_{t,k+1}$ by solving the following Sylvester equation
    $(\mathbf{K} + \lambda\mathbf{I})\mathbf{B} + 2\rho\mathbf{B}\mathbf{L}_t = \mathbf{K} + \gamma\mathbf{C}_k - \mathbf{M}_k$;
    Get $\mathbf{C}_{k+1} = \arg\min_{\mathbf{C}} Q(\mathbf{B}_{t,k+1}, \mathbf{C}, \mathbf{M}_k)$;
    $\mathbf{M}_{k+1} = \mathbf{M}_k + \gamma(\mathbf{B}_{t,k+1} - \mathbf{C}_{k+1})$;
    $k = k + 1$;
  until $\|\mathbf{B}_{t,k} - \mathbf{C}_k\|_F^2 < \epsilon_2$;
  $\mathbf{B}_{t+1} = \mathbf{B}_{t,k}$;
  Calculate new $\mathbf{W}_{t+1} = (|\mathbf{B}_{t+1}| + |\mathbf{B}_{t+1}^T|)/2$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2/n^2 < \epsilon_1$;
Get final Kernel SGLS graph $\mathbf{W} = \mathbf{W}_t$.

Algorithm 7: Kernel SGLS graph construction

Since $\mathbf{B}$ is asymmetric, its Laplacian matrix will be given by: $\mathbf{L_B} = \mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T$. Then the above criterion becomes:

$$\begin{aligned} &\min \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 \\ &\quad + \rho\,\mathrm{Tr}(\mathbf{B}(\mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T)\mathbf{B}^T), \\ &s.t.\, B_{ij} \geq 0 \,\text{and}\, B_{ii} = 0, \end{aligned} \tag{5.10}$$

where $\mathbf{D}_r$ and $\mathbf{D}_c$ are two diagonal matrix whose elements are the row sums and the column sums of $\mathbf{B}$, respectively.

Note that minimizing Eq.(5.10) is subject to $B_{ij} \geq 0$. Let $\mathbf{Z}$ with $Z_{ij} \geq 0$ be the corresponding Lagrange multipliers. Consider the Lagrange function $F(\mathbf{B})$ as:

$$\begin{aligned} F(\mathbf{B}) =& \|\Phi - \Phi\,\mathbf{B}\|_F^2 + \lambda\,\|\mathbf{B}\|_1 \\ &+ \rho\,\mathrm{Tr}(\mathbf{B}(\mathbf{D}_r + \mathbf{D}_c - \mathbf{B} - \mathbf{B}^T)\mathbf{B}^T) + \mathrm{Tr}(\mathbf{Z}\,\mathbf{B}^T). \end{aligned} \tag{5.11}$$

It's easy to show that we have $\mathrm{Tr}(\mathbf{Z}\,\mathbf{B}^T) = \sum Z_{ij}B_{ij}$.

Then taking the derivative of $F(\mathbf{B})$ with respect to $\mathbf{B}$ leads to:

$$\begin{aligned} \frac{\partial F(\mathbf{B})}{\partial \mathbf{B}} =& 2\mathbf{K}\mathbf{B} - 2\mathbf{K} + \lambda\mathbf{E} + \rho(\mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B})) \\ &- 2\rho(\mathbf{B}\mathbf{B} + \mathbf{B}^T\mathbf{B} + \mathbf{B}\mathbf{B}^T) + \mathbf{Z}, \end{aligned} \tag{5.12}$$

where $\mathbf{K} = \Phi^T\Phi$. One suggestion for $\mathbf{K}$ is Gaussian Kernel, i.e. $K_{ij} = e^{-\|\mathbf{x}_i-\mathbf{x}_j\|^2/\sigma^2}$. Every element of $\mathbf{E}$ is 1, and $\mathbf{M}_r(\mathbf{B}) = \partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)/\partial\mathbf{B}$, $\mathbf{M}_c(\mathbf{B}) = \partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)/\partial\mathbf{B}$.

It's difficult to have simple expressions for $\mathbf{M}_r(\mathbf{B})$ and $\mathbf{M}_c(\mathbf{B})$. But we can have a detailed look into every elements of them. Let $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$. Then we can have $\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T) = \sum_k D_{r(k)}\|\mathbf{b}_k\|^2$, where $D_{r(k)}$ is the k-th diagonal elements of $\mathbf{D}_r$, i.e. $D_{r(k)} = \sum_t B_{kt}$. Similarly, $\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T) = \sum_k D_{c(k)}\|\mathbf{b}_k\|^2$, where $D_{c(k)} = \sum_t B_{tk}$.

Then the derivative of $\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)$ and $\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)$ with respect to $Bvect_{ij}$ can be obtained:

$$\frac{\partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)}{\partial B_{ij}} = \|\mathbf{b}_i\|^2 + 2B_{ij}D_{r(j)}, \tag{5.13}$$

and

$$\frac{\partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)}{\partial B_{ij}} = \|\mathbf{b}_j\|^2 + 2B_{ij}D_{c(j)}. \tag{5.14}$$

With these, every element of $\mathbf{M} = \mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B})$ can be obtained:

$$\begin{aligned} M_{ij} &= (\mathbf{M}_r(\mathbf{B}) + \mathbf{M}_c(\mathbf{B}))_{ij} \\ &= \frac{\partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_r\mathbf{B}^T)}{\partial B_{ij}} + \frac{\partial\mathrm{Tr}(\mathbf{B}\mathbf{D}_c\mathbf{B}^T)}{\partial B_{ij}} \\ &= \|\mathbf{b}_i\|^2 + \|\mathbf{b}_j\|^2 + 2B_{ij}D_{r(j)} + 2B_{ij}D_{c(j)}. \end{aligned} \tag{5.15}$$

Then $\partial F(\mathbf{B})/\partial Bij$ can be:

$$\begin{aligned} \frac{\partial F(\mathbf{B})}{\partial B_{ij}} =& (2\mathbf{K}\mathbf{B} - 2\mathbf{K} + \lambda\mathbf{E} - 2\rho(\mathbf{B}\mathbf{B} + \mathbf{B}^T\mathbf{B} + \mathbf{B}\mathbf{B}^T))_{ij} \\ &+ Z_{ij} + \rho M_{ij}. \end{aligned} \tag{5.16}$$

With $\partial F(\mathbf{B})/\partial\mathbf{B} = 0$ and the *Karush-Kuhn-Tucker* (KKT) condition $Z_{ij}B_{ij} = 0$, we will have:

$$(2\mathbf{K}\mathbf{B} - 2\mathbf{K} + \lambda\mathbf{E} - 2\rho(\mathbf{B}\mathbf{B} + \mathbf{B}^T\mathbf{B} + \mathbf{B}\mathbf{B}^T) + \rho\mathbf{M})_{ij}B_{ij} = 0. \tag{5.17}$$

An iteratively updating rule on $\mathbf{B}$ is designed as

$$B_{ij} \leftarrow \frac{2K_{ij} + 2\rho(\mathbf{B}\mathbf{B} + \mathbf{B}^T\mathbf{B} + \mathbf{B}\mathbf{B}^T)_{ij}}{(2\mathbf{K}\mathbf{B} + \lambda\mathbf{E} + \rho\mathbf{M})_{ij}}B_{ij}. \tag{5.18}$$

Thus, we can use an iterative algorithm. Note the initial $\mathbf{B}_0$ will be important since when B_{ij} will maintain 0 when it once become 0 in the iteration. One simple choice of the initial $\mathbf{B}_0$ is a matrix with all the elements is 1 except the diagonal elements is 0. In other word, $\mathbf{B}_0 = \mathbf{E} - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

The steps of the proposed method are described in Algorithm 8.

Input: The original data matrix $\mathbf{X}$, initial matrix $\mathbf{B}_0$ λ, ρ and ϵ (a small positive threshold)
Output: DSGLS graph $\mathbf{W}$

Set $t = 0$;
Calculate the kernel matrix $\mathbf{K}$;
repeat
  Calculate matrix $\mathbf{M}_t$ using Eq.(5.15);
  Update matrix $\mathbf{B}$ with updating rule
  $B_{ij} \leftarrow \frac{2K_{ij}+2\rho(\mathbf{B}_t\mathbf{B}_t+\mathbf{B}_t^T\mathbf{B}_t+\mathbf{B}_t\mathbf{B}_t^T)_{ij}}{(2\mathbf{K}\mathbf{B}_t+\lambda\mathbf{E}+\rho\mathbf{M}_t)_{ij}} B_{ij}$;
  $t = t + 1$;
until $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|_F^2/n^2 < \epsilon$;
Get DSGLS graph $\mathbf{W} = (\|\mathbf{B}\| + \|\mathbf{B}^T\|)/2$.

Algorithm 8: DSGLS graph construction

5.4 Performance evaluation

In general, the estimated graph alone cannot lend itself to an easy assessment of the method that constructs it. Indeed, given a real data set as well as a machine learning task that uses this data set, it is very often challenging if not unfeasible to know in advance the ideal graph for that data set and for that task. Thus, in our work, the graph-construction methods are assessed by the performance of the post-graph construction tasks, i.e. two label propagation methods (GRF and LPDGL) and two dimensionality reduction methods: one non-linear dimensionality reduction method (LE) and one linear dimensionality reduction method (LPP). In this section, we evaluate the proposed graph construction methods in the main application of computer vision: face recognition. We use semi-supervised label propagation based on the built graph. The performance will be mainly given by recognition and classification accuracy. The construction methods used for comparison are kNN graph [5], LLE graph [88], standard sparse graph (ℓ_1-s) [65], robust sparse graph (ℓ_1-r) [94], and constrained sparse graph (ℓ_1-c) [19].

In this section, we evaluate the proposed algorithm on several real databases. To this end, five face databases are used: ORL, FERET, Extended Yale B, PF01 and PIE.

5.4.1 Comparison among several graph construction methods

To evaluate the performance of our proposed methods, we compare them with several other graph construction methods including KNN graph, LLE graph, standard sparse graph, robust sparse graph and constrained sparse graph. For every graph construction method, several values for the parameter are used. We then report the top-1 recognition accuracy (best average recognition rate) of all methods from the best parameter configuration.

kNN and LLE methods have the neighborhood size parameter k. The robust sparse graph has the parameter λ and the constrained sparse graph has α and β to set. The two proposed methods have parameters λ and ρ, the initial $\mathbf{W}_0$ and the threshold ϵ_1 and ϵ_2 to set. In our experiments, k is chosen from 3 to 60 with a step of 3 for kNN and LLE graph construction methods; $\lambda = 0.1$ for robust graph and $\alpha = \beta = 1$ for the constrained sparse graph; for the proposed methods, λ is chosen from $\{10^{-4}, 10^{-3}, 0.01, 0.1\}$ and ρ is chosen from $\{10^{-4}, 10^{-3}, 0.01, 0.1, 1\}$, $\mathbf{W}_0$ is set by the kNN graph when $k = 3$ and $\epsilon_1 = \epsilon_2 = 10^{-3}$. For the kernel version of SGLS, we use the Gaussian Kernel $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$. The parameter of the Gaussian is set to the average distances among the samples.

After we construct the graphs on the original data, we run the label propagation method GRF and LPDGL as classifiers. For LPDGL, β and γ are set to 1. For label propagation, we randomly split the whole data set into a labeled part and unlabeled part and repeat this process 10 times. The final performance (recognition rate) is given by the average.

Tables 5.1, 5.2, 5.3, 5.4, and 5.5 illustrate the method comparison obtained with the GRF label propagation method on five data sets.

Table 5.1: Average recognition rates (%) on ORL by GRF.

Method\l	1	2	3	4	5
kNN [5]	75.1	85.2	89.6	92.3	94.2
LLE [88]	74.1	83.7	88.6	93.1	96.2
ℓ_1-s [65]	76.8	86.7	91.4	93.9	95.6
ℓ_1-r [94]	64.8	80.4	87.6	91.2	93.8
ℓ_1-c [19]	78.9	87.2	91.9	**94.4**	96.1
SGLS	79.3	88.0	92.0	94.4	96.3
CKSGLS	**82.3**	**88.8**	**92.9**	**94.7**	**96.5**
KSGLS	74.5	84.8	91.5	94.0	96.1

Table 5.1 shows the mean recognition rates over the 10 random splits on ORL with different graph construction methods. In this table, different label numbers l are used. On ORL, the DSGLS method outperforms the other methods.

Table 5.2 shows the average recognition rates on FERET with different graph construction methods. In the table, the number of labeled samples varies from 1 to 5 samples per class. On FERET, the SGSL method and kernelized version out-perform the other graph construction methods and the kernelized SGSL method performs best.

Table 5.3 shows the mean recognition rates on Extended Yale B with different graph construction methods. Several label percentages are used in the interval 10%-30%. For Extended Yale B data set, the SGSL method and

Table 5.2: Average recognition rates (%) on FERET by GRF.

Method\l	1	2	3	4	5
kNN	22.4	30.9	39.0	45.2	51.0
LLE	31.3	43.9	55.7	66.9	72.7
ℓ_1-s	40.2	56.0	66.5	72.7	74.8
ℓ_1-r	39.1	57.3	70.6	79.3	81.4
ℓ_1-c	37.0	51.2	61.0	68.8	72.5
SGLS	42.8	61.4	74.2	80.6	82.0
DSGLS	44.5	61.4	72.3	79.5	82.1
KSGLS	**49.6**	**67.8**	**78.0**	**84.6**	**86.0**

Table 5.3: Average recognition rates (%) on Extended Yale B by GRF.

Method\l	10%	20%	30%
kNN	81.7	89.1	84.1
LLE	66.3	74.9	89.3
ℓ_1-s	79.3	86.6	90.0
ℓ_1-r	90.5	94.0	96.1
ℓ_1-c	75.7	83.7	87.3
SGLS	**92.7**	**95.3**	**96.8**
DSGLS	88.4	92.0	93.9
KSGLS	91.0	94.1	95.6

Table 5.4: Average recognition rates (%) on PF01 by GRF.

Method\l	2	4	6	8	10
kNN	35.5	43.3	47.0	50.6	53.8
LLE	34.6	43.6	51.2	64.9	73.7
ℓ_1-s	40.3	50.4	55.9	59.3	63.6
ℓ_1-r	57.5	69.7	75.6	80.1	86.0
ℓ_1-c	39.5	50.4	55.7	60.7	65.6
SGLS	61.7	72.9	79.7	84.0	88.8
DSGLS	47.7	62.1	67.8	72.5	78.5
KSGLS	**61.8**	**74.7**	**80.5**	**85.2**	**90.7**

Table 5.5: Average recognition rates (%) on PIE by GRF.

Method\l	2	4	6	8	10
kNN	22.5	32.0	40.0	43.6	46.6
LLE	17.1	25.8	36.5	40.6	46.8
ℓ_1-s	38.8	49.2	57.1	62.0	67.9
ℓ_1-r	51.4	68.0	76.8	81.9	85.7
ℓ_1-c	29.6	40.5	50.1	54.9	59.2
SGLS	**52.8**	**69.3**	**78.2**	**83.1**	**86.7**
DSGLS	37.1	51.1	61.4	66.5	70.2
KSGLS	42.5	55.7	65.1	70.9	77.3

kernelized version out-perform the other graph construction methods and the SGSL method performs best.

Table 5.4 shows the mean recognition rates on PF01. Several label numbers are used (2-10 samples). On PF01, the SGSL method and kernelized version out-perform the other graph construction methods and the kernelized SGSL method performs best.

Table 5.5 shows the mean recognition rates on PIE. Several label numbers are used (2-10 samples). On PIE, the SGLS method outperforms the other methods.

Tables 5.6, 5.7, and 5.8 illustrate the method comparison obtained with the LPDGL label propagation method on Extended Yale B, PF01 and PIE.

Table 5.6: Average recognition rates (%) on Extended Yale B by LPDGL.

Method\l	10%	20%	30%
kNN	81.3	83.7	85.1
LLE	84.5	86.2	87.8
ℓ_1-s	84.8	91.5	94.9
ℓ_1-r	83.2	89.1	91.7
ℓ_1-c	81.1	87.0	89.5
SGLS	**90.1**	**94.2**	**96.2**

Table 5.6 shows the mean recognition rates on Extended Yale B with different graph construction methods when the label propagation method is LPDGL. Several label percentages are used in the interval 10%-30%. For Extended Yale B data set, the SGSL method performs best.

Table 5.7 shows the mean recognition rates on PF01 when the label propagation method is LPDGL. Several label numbers are used (2-10 samples). On PF01, the SGSL method out-performs the other graph construction methods.

Table 5.7: Average recognition rates (%) on PF01 by LPDGL.

Method\l	2	4	6	8	10
kNN	33.9	42.0	46.1	49.5	53.4
LLE	34.6	43.9	50.3	55.0	61.9
ℓ_1-s	49.9	65.2	72.7	78.0	84.8
ℓ_1-r	49.4	57.8	61.5	63.9	67.0
ℓ_1-c	44.7	55.0	59.9	64.4	69.6
SGLS	**62.4**	**74.2**	**79.3**	**83.8**	**88.8**

Table 5.8: Average recognition rates (%) on PIE by LPDGL.

Method\l	2	4	6	8	10
kNN	19.9	28.8	36.3	39.0	41.7
LLE	22.6	31.6	40.1	43.3	46.2
ℓ_1-s	44.3	61.4	72.2	78.4	82.7
ℓ_1-r	41.5	51.0	58.5	62.6	66.2
ℓ_1-c	33.7	43.5	51.5	55.6	59.9
SGLS	**51.3**	**67.0**	**76.1**	**81.5**	**85.7**

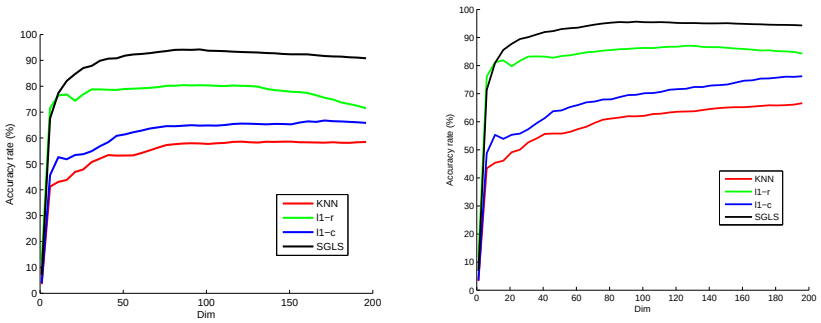
Table 5.8 shows the average recognition rates on PIE when the label propagation method is LPDGL. Several label numbers are used (2-10 samples). On PIE, the SGLS method outperforms the other methods.

Analysis of results: According to the results reported in the previous tables, we can make the following observations:

- In general, the proposed methods (including the DSGLS method and the kernelized version) outperform the other methods.
- The proposed SGLS method is almost always superior to the robust sparse graph method suggesting that Laplacian smoothness has contributed to get more informative graphs.
- The proposed SGLS method can improve the performance significantly for big data sets (like PF01 and PIE) even when the number of the labeled samples is small.
- The proposed SGLS and DSGLS methods can out-perform the other graph construction method under different label propagation method.
- From some datasets, the direct method DSGLS has not provided very good results. This can be explained by the fact that the kernel used was fixed. Possibly, one can expect that with other kernel types performance of the direct method can be improved. The same observation can be also true for the behavior of the kernelized version of SGLS (KSGLS).

5.4.2 SGLS based Laplacian Eigenmaps

Laplacian Eigenmaps (LE) is a non-linear dimensionality reduction method [4]. In this section, we use different graphs as the $\mathbf{W}$ in LE method for dimensionality reduction and then for classification. We construct graphs on the whole data using kNN, robust sparse graph, constrained sparse graph and the proposed SGLS method. Then the graphs are used for LE and nearest classifier will be employed for classification in a range dimension from 1 to 200. 15% and 31% samples every class are randomly labeled for Extended Yale B, while 30% and 70% samples are randomly chosen for label for PF01 and PIE. The experiment repeats 10 times.



(a) 15% labeled (b) 31% labeled

Fig. 5.1: Average recognition rate variation different graphs based LE on Extended Yale B (15% and 31% labeled).

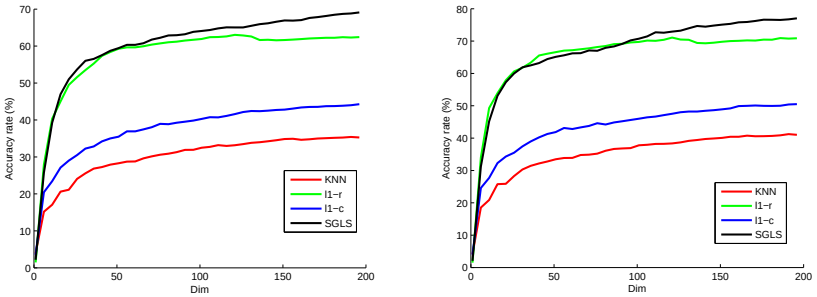
Figures 5.1, 5.2 and 5.3 show the average recognition rates with different graphs. According to the figures shown and the other results we have, it can be drawn that the graph constructed by the SGLS method performs best in LE among the other graph construction methods.

5.4.3 SGLS based Locality Preserving Projection

Locality Preserving Projection [36] can be regarded as a linearized version of LE.

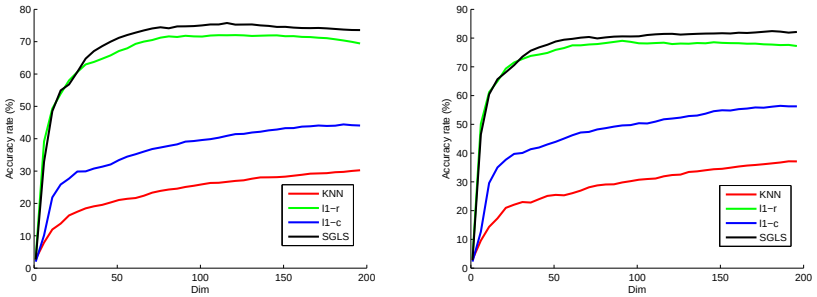
In this section, we use different graphs as the $\mathbf{W}$ in LPP method for dimensionality reduction and then for classification. We choose different percentages (15%, 23% and 31% for Extended Yale B and 30%, 50% and 70% for PF01 and PIE) of data as training set and the rest testing set. We perform LPP on the training set and then apply nearest neighborhood classifier for classification.

Table 5.9, 5.10 and 5.11 show the recognition rates using LPP with different graphs.



(a) 30% labeled (b) 70% labeled

Fig. 5.2: Average recognition rate variation different graph based LE on PF01 (30% and 70% labeled).



(a) 30% labeled (b) 70% labeled

Fig. 5.3: Average recognition rate variation different graph based LE on PIE (30% and 70% labeled).

Table 5.9: LPP evaluation on Extended Yale B.

W	15%	23%	31%
kNN	49.0	58.3	66.1
ℓ_1-r	54.9	59.8	65.0
ℓ_1-c	49.3	51.3	58.8
SGLS	**74.4**	**79.4**	**80.3**

Table 5.10: LPP evaluation on PF01.

W	30%	50%	70%
kNN	32.3	41.7	32.9
ℓ_1-r	34.9	43.0	31.4
ℓ_1-c	35.2	42.8	30.1
SGLS	**43.8**	**54.6**	**40.4**

Table 5.11: LPP evaluation on PIE.

W	30%	50%	70%
kNN	29.7	42.3	46.8
ℓ_1-r	36.6	44.4	49.3
ℓ_1-c	26.6	38.4	38.9
SGLS	**41.5**	**56.1**	**61.8**

According to the tables, the SGLS method can out-perform the other graph construction methods in LPP.

5.4.4 Stability of the proposed method

In this section, we will empirically show that independently of the initial weight matrix $\mathbf{W}_0$ used by the Laplacian smoothness criterion, the SGLS method will provide the same graph. To this end, we conduct the following experiment. We initialize the SGLS method with different weight matrices $\mathbf{W}_0$. Here we use the kNN graph, the robust sparse graph and a random one for initial graphs. For each such initialization, we run the iterative SGLS method in order to compute the optimal $\mathbf{B}$. We then compare the obtained graphs as well as performance obtained with them. To evaluate the trend of Laplacian smoothness term, we calculate the values of $\mathrm{Tr}(\mathbf{BLB}^T)$ in every iteration with a fixed $\mathbf{L}$ (here we use the kNN graph, the robust ℓ_1 graph and the SGLS graph in every iteration to calculate the Laplacian matrix $\mathbf{L}$).

In this comparison, we use the performance of the GRF label propagation method based on the affinity matrices $\mathbf{B}$ obtained at convergence (for all types of initialization). We also calculate the differences between the $\mathbf{B}$'s for every iteration and the differences between the optimal $\mathbf{B}$ computed with different initial weight matrices. We set $\lambda = 0.01$, $\rho = 0.1$ and 3 labeled samples for FERET and 6 labeled samples for PF01.

Table 5.12, 5.13 and 5.14 show the obtained results on FERET and Table 5.15, 5.16 and 5.17 show the results on PF01. Table 5.12 and Table 5.15 show the average recognition rates associated with the affinity matrix $\mathbf{B}$ obtained at each iteration of the SGLS method. In each Table, three different initial weight matrices are used.

Table 5.13 and Table 5.16 illustrate the difference between the graphs obtained in two consecutive iterations $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$, that is the Frobenius norm $\|\mathbf{B}_t - \mathbf{B}_{t-1}\|^2$, $\mathbf{B}_0 = \mathbf{W}_0$. Table 5.14 and Table 5.17 show the difference between the optimal $\mathbf{B}$s obtained by different initial weight matrices $\mathbf{W}_0$. Here, $\mathbf{B}^*_{knn}$ refers to the optimal $\mathbf{B}$ obtained by SGLS when the initial weight matrix is the kNN. $\mathbf{B}^*_{\ell_1}$ refers to the optimal $\mathbf{B}$ obtained by SGLS when the initial weight matrix is set to the robust sparse graph (ℓ_1-r). $\mathbf{B}^*_{rand}$ refers to the optimal $\mathbf{B}$ obtained by SGLS when the initial weight matrix is set to completely random graph.

Table 5.12: Average recognition rates (%) of each $\mathbf{B}_t$ in iteration on FERET with 3 labeled samples per class.

Initial $\mathbf{W}_0$\Iteration (t)	0	1	2	3	4
kNN	28.2	72.5	74.1	74.1	74.1
ℓ_1-r	69.0	75.1	74.2	74.1	74.1
random	0.7	73.3	74.0	74.1	74.1

Table 5.13: Differences between $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$ in iteration on FERET.

Initial $\mathbf{W}_0$	ϵ_1	ϵ_2	ϵ_3	ϵ_4
kNN	1.8e4	4.9	5.8e-2	1.0e-3
ℓ_1-r	2.4e2	7.1	1.0e-2	2.8e-4
random	5.7e5	1.2	3.5e-2	1.2e-3

Table 5.14: Differences between optimal $\mathbf{B}$s on FERET.

$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{\ell_1}\|^2$	$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{rand}\|^2$
2.3e-6	2.7e-6

According to the recognition rates in Table 5.12 and Table 5.15, we can see that in less than 5 iterations, the optimal $\mathbf{B}$ can be obtained. From the results depicted in the above tables, we can conclude that regardless of the initialization used, the SGLS is able to estimate the same optimal graph. In particular, Table 5.14 and Table 5.17 show that different initial $\mathbf{W}_0$ will lead to the same optimal SGLS graph $\mathbf{B}$. This fact makes the SGLS not dependent on a given initialization.

Table 5.15: Average recognition rates (%) of $\mathbf{B}_t$ in iteration on PF01 with 6 labeled every class.

$\mathbf{W}_0$\Iteration (t)	0	1	2	3	4
kNN	40.0	74.6	77.1	77.3	77.3
ℓ_1-r	75.5	78.1	77.4	77.4	77.3
random	1.1	76.9	77.2	77.3	77.3

Table 5.16: Differences between $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$ in iteration on PF01.

Initial $\mathbf{W}_0$	ϵ_1	ϵ_2	ϵ_3	ϵ_4
kNN	2.2e4	4.2	6.1e-2	1.4e-3
ℓ_1-r	3.1e2	1.6	4.8e-2	2.2e-3
random	9.6e5	1.5	5.6e-2	2.7e-3

Table 5.17: Differences between optimal $\mathbf{B}$s on PF01.

$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{\ell_1}\|^2$	$\|\mathbf{B}^*_{knn} - \mathbf{B}^*_{rand}\|^2$
8.8e-4	1.9e-3

As illustrated in Table 5.13 and Table 5.16, the proposed methods will be convergence within 5 iterations. Actually, the SGLS algorithm can almost obtain the optimal $\mathbf{B}$ in 2 iterations.

Finally, Table 5.18 illustrates the evolution of the smoothness term, i.e. the third term in the criterion $\mathrm{Tr}(\mathbf{BLB}^T)$ when the Laplacian matrix is fixed and the graph asymmetric matrix $\mathbf{B}$ is estimated by minimizing Eq.(5.3). The results correspond to five data sets and to three graphs: kNN graph, ℓ_1 graph and SGLS graph $\mathbf{B}_t$ in every iteration. From this Table, we can see that the value of $\mathrm{Tr}(\mathbf{BLB}^T)$ decreases by the successive minimizations which indicates that the Laplacian smoothness term of the estimated graph is really decreasing over over the recursions.

5.4.5 Sensitivity to parameters

In the SGLS algorithm, parameters λ and ρ should be set. We aim to study the recognition rates obtained by SGLS when these two parameters vary. Figure 5.4 illustrates the recognition rates with different parameter values for FERET, Extended Yale B, PF01 and PIE.

On FERET, 5 samples for each person are labeled. According to Fig.5.4a, $\lambda = 0.01$ and $\rho = 0.1$ will lead to a better graph for final recognition rates.

On Extended Yale B, 13 samples for each person are labeled. According to Fig.5.4b, $\lambda = 0.1$ and $\rho = 0.1$ will lead to a better graph for final recognition rates.

Table 5.18: The value of $Tr(\mathbf{B}_t\mathbf{L}\mathbf{B}_t^T)$ in each iteration.

Data sets	$\mathbf{L}$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
ORL	kNN	5915.1	67.5	84.7	90.0	91.6
	ℓ_1-r	5627.3	61.0	71.3	74.9	76.0
	$\mathbf{L}_t$	5915.1	64.6	69.3	70.9	71.3
FERET	kNN	28588.0	191.4	205.3	207.8	208.1
	ℓ_1-r	32203.1	227.3	219.3	219.8	219.9
	$\mathbf{L}_t$	28588.0	214.1	204.1	204.0	204.0
Extended Yale B	kNN	30909.1	132.8	136.0	137.0	137.2
	ℓ_1-r	34913.7	176.3	169.6	169.6	169.7
	$\mathbf{L}_t$	30909.1	162.0	154.8	154.5	154.5
PF01	kNN	32888.7	131.5	135.5	136.3	136.3
	ℓ_1-r	40770.5	195.5	178.7	177.3	177.0
	$\mathbf{L}_t$	32888.7	165.3	152.3	151.3	151.1
PIE	kNN	40009.3	232.6	253.1	257.8	259.0
	ℓ_1-r	45625.0	269.4	265.9	268.2	269.1
	$\mathbf{L}_t$	40009.3	256.0	246.3	246.9	246.5

On PF01, 10 samples for each person are labeled. According to Fig.5.4c, $\lambda = 0.01$ and $\rho = 0.1$ will lead to a better graph for final recognition rates.

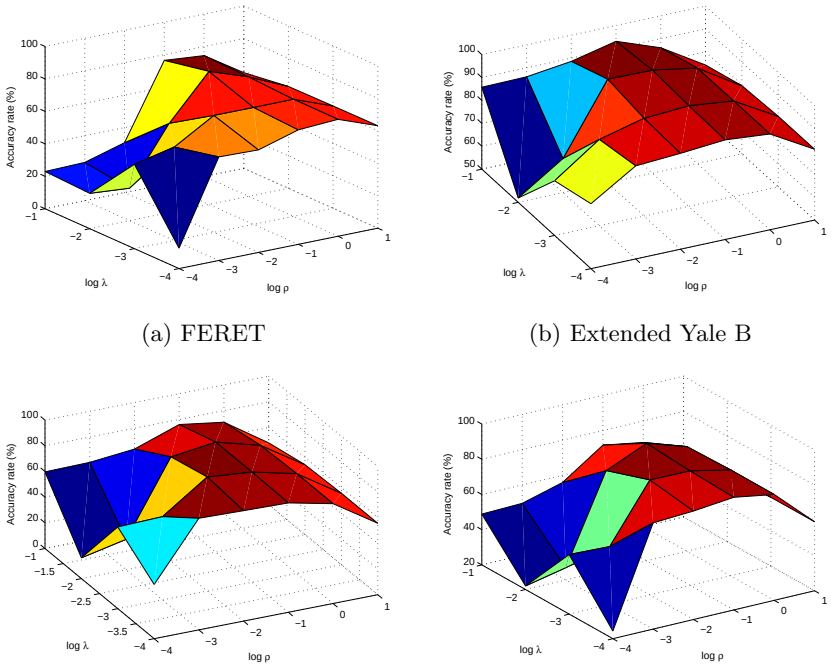
On PIE, 10 samples for each person are labeled. According to Fig.5.4d, $\lambda = 0.01$ and $\rho = 0.1$ will lead to a better graph for final recognition rates.

Also the other results in our experiments show that the optimal can be reached when $\lambda \in \{0.01, 0.1\}$ and $\rho \in \{0.01, 0.1\}$.

5.5 Conclusion

In this chapter, we proposed a sparse graph construction method using Laplacian smoothness as a constraint. The proposed method preserves the smoothness of the sparse representation matrix. Moreover we introduce kernel version of SGLS and a direct solution to the SGLS criterion. According to the experimental results, the proposed methods can handle real data sets with a better performance than other graph construction methods in general. The proposed SGLS method is more efficient but can perform as well as and even better than the other graph construction methods.

One of the disadvantages of the proposed method is the choice of parameters. Although the experiments show that the parameters λ and ρ in the proposed criterion should be around 0.01 and 0.1, still no evidences show that the parameters can be decided directly.



(a) FERET (b) Extended Yale B

(c) PF01 (d) PIE

Fig. 5.4: Recognition rate as a function of parameter λ and ρ on FERET, Extended Yale B, PF01 and PIE.

Part II

Semi-supervised Embedding

6

Advances in semi-supervised embedding

Abstract

In this chapter we will have a brief introduction about semi-supervised learning and then describe three techniques that represent the state-of-the art in graph-based semi-supervised embedding.

Contents

6.1 Introduction

In many real world applications, such as face recognition and text categorization, the data are usually provided in a high dimension space. Learning from the high-dimensional data becomes the primary problem in the area of machine learning and data mining technology. Dimensionality learning can overcome the problem of "curse of dimensionality". In general, dimensionality reduction methods can be classified into feature selection and feature extraction [29]. Feature selection reduces the dimension of the original data by selecting fewer features which can well represent the original, have distinguishing ability and also should be convenient for the follow-up learning tasks [33]. Feature extraction (dimension reduction or manifold learning), also regarded as embedding, reduces the dimension of the original data by combining multiple features linearly and nonlinearly while preserving the structural information within the original data [84]. How to extract features effectively and provide the best feature subset for subsequent learning tasks (such as classification, clustering, etc.) becomes the key to learning a successful model.

A lot of supervised and unsupervised embedding methods for dimension reduction have been proposed. Principal Component Analysis (PCA) [93] and Multidimensional Scaling (MDS) [8] are two classic linear unsupervised embedding methods. PCA tries to obtain the best projection direction of the data by maximizing the covariance of the data. PCA is widely used in face recognition and other application tasks because of its high efficiency and simple calculation. Linear Discriminant Analysis (LDA) [2, 18] is a classic supervised method for dimensionality reduction which seeks the best projection direction by maximizing the intra-class covariance while minimizing inter-class covariance. LDA is also widely used in face recognition, font recognition and speech recognition due to its effectiveness and intuition. Both PCA and LDA mainly study the linear projection in the high-dimensional space and their advantages include simple computation and can produce a simple transformation function. They are effective for high-dimensional data in linear structures but can rarely well perform for non-linear high-dimensional data.

Most of high-dimensional data in the real-world are nonlinear which makes it difficult to find out the geometric structure and correlation of the high-dimensional data and reveal the manifold distribution by using the above linear methods. In recent years, many non-linear manifold learning methods have been proposed for the nonlinear properties of the high-dimensional data. In 2000, Locally Linear Embedding [66] (LLE) and Isometric Feature Mapping (ISOMAP) [82] were separately proposed in *science* which laid a foundation of manifold learning. Soon afterward, M. Belkin *et al.* proposed Laplacian Eigenmaps [5] (LE). And lots of manifold learning methods are proposed [110, 92, 70, 43]. However these nonlinear manifold learning methods are transductive which can hardly deal with the out-of-sample problem, i.e. they can not provide a direct embedding for new data samples.

He *et al.* proposed both Locality Preserving Projection (LPP) [36], essentially a linearized version of LE, and Neighborhood Preserving Embedding [34] (NPE), a linearized version of LLE. LPP and NPE can be interpreted in a general graph embedding framework with different choices of graph structure. Cai et al. proposed Spectral Regression (SR) [11] which avoid the computation of eigenvalues. Moreover, most of the linear methods can be extended to nonlinear variants using kernel techniques, such as kernel PCA (KPCA) [67], kernel LDA (KDA) [48].

Afterwards, sparse representation [45, 97, 65] based methods have attracted extensive attention. Lai *et al.* proposed a 2-D feature extraction method called sparse 2-D projections for image feature extraction [44]. In [46], a robust tensor learning method called sparse tensor alignment (STA) is then proposed for unsupervised tensor feature extraction based on the alignment framework. In [47], multilinear sparse principal component analysis (MSPCA) inherits the sparsity from the sparse PCA and iteratively learns a series of sparse projections that capture most of the variation of the tensor data.

In many real-world problems, collecting a large number of labeled samples is practically impossible. The reason is twofold. Firstly, these labeled samples

can be very few. Secondly, acquiring labels requires expensive human labor. To deal with this problem, semi-supervised embedding methods can be used to project the data in the high-dimensional space into a space with fewer dimensions. In the last decade, semi-supervised learning algorithms have been developed to effectively utilize a large amount of unlabeled samples as well as the limited number of labeled samples for real world applications [112, 55, 113, 54, 99, 21, 50, 78, 49].

Cai et al. proposed semi-supervised discriminant analysis (SDA) [10]. Since SDA is sensitive to the noise, Zhang et al. proposed Semi-Supervised Discriminant Analysis (SSDA) [109]. Sugiyama et al. proposed Semi-supervised Local Fisher discriminant analysis (SELF) [77] which has an analytic form of the globally optimal solution and it can be computed based on eigendecomposition. Qiao et al. proposed Sparsity preserving discriminant analysis (SPDA) [64].

In the past years, many graph-based methods for semi-supervised learning have been developed [106, 13, 1, 100, 52, 95, 56, 62, 39, 107, 27, 108, 83]. Some edge based (constraints of sample pairs) methods are proposed. Zhang et al. proposed Semi-Supervised Dimensionality Reduction (SSDR) [103] which can preserve the intrinsic structure of the unlabeled data as well as both the must-link and cannot-link constraints defined on some examples in the projected low-dimensional space. Wei et al. proposed neighbourhood preserving based semi-supervised dimensionality reduction algorithm [91]. Cevikalp proposed Constrained Locality Preserving Projections (CLPP) [14] that incorporates pairwise equivalence constraints for finding a better embedding space. Zhang et al. proposed two pairwise constraints preserving projection methods [102] by using the bagging [9] and boosting [31] techniques.

Constrained Laplacian Eigenmaps [17] (CLE) is a semi-supervised embedding method. CLE constrains the solution space of Laplacian Eigenmaps only to contain embedding results that are consistent with the labels. Labeled points belonging to the same class are merged together, labeled points belonging to different classes are separated, and similar points are close to one another. Similarly, Constrained Graph Embedding (CGE) [35] tries to project the data points from a same class onto one single point in the projection space with a constraint matrix.

Flexible Manifold Embedding (FME) [57] is a label propagation method. FME simultaneously estimates the non-linear embedding of unlabeled samples and the linear regression over these non-linear representations. In [30], the authors propose a whole learning process that can provide the data graph and a linear regression within a same framework.

6.2 Graph-based semi-supervised embedding methods

In this section, we introduce several existing graph-based semi-supervised embedding methods including Sparsity preserving discriminant analysis (SP-

DA), Semi-supervised Discriminant Embedding (SDE) [101] and Constrained Graph Embedding (CGE).

Some mathematical notations are listed and will be used in the next several sections. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \Re^{m \times n}$ be the data matrix, where n is the number of training samples and m is the dimension of each sample. Let $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ be a one-dimensional map of $\mathbf{X}$. Under a linear projection $\mathbf{y}^T = \mathbf{p}^T\mathbf{X}$, each data point $\mathbf{x}_i$ in the input space $\Re^m$ is mapped into $y_i = \mathbf{p}^T\mathbf{x}_i$ in the real line. Here $\mathbf{p} \in \Re^m$ is a projection axis. Let $\mathbf{Y} \in \Re^{d \times n}$ be the data projections in a d dimensional space.

6.2.1 Locality Preserving Projection

Locality Preserving Projection [36] (LPP) is a classic unsupervised embedding method which aims to preserve the local structure of the data by keeping two sample points close in the projection space when they are similar in the original space. The reasonable criterion of LPP is to optimize the following objective function under some constraints:

$$\min \sum_{i,j} (y_i - y_j)^2 W_{ij}, \tag{6.1}$$

where $\mathbf{W}$ is the affinity matrix associated with the data and W_{ij} represents the similarity between sample $\mathbf{x}_i$ and sample $\mathbf{x}_j$. Estimating the graph affinity $\mathbf{W}$ from data can be carried out by many graph construction methods [74]. The simplest method is based on the use of kNN graph as defined in Eq.(3.1).

After some simple algebraic formulations, we obtain:

$$\sum_{i,j} (y_i - y_j)^2 W_{ij} = 2\mathbf{p}^T\mathbf{X}\,\mathbf{L}\,\mathbf{X}^T\mathbf{p}, \tag{6.2}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix and $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$.

With the constraint $\mathbf{p}^T\mathbf{X}\,\mathbf{D}\,\mathbf{X}^T\mathbf{p} = 1$, the problem becomes:

$$\min_{\mathbf{p}} \frac{\mathbf{p}^T\mathbf{X}\,\mathbf{L}\,\mathbf{X}^T\mathbf{p}}{\mathbf{p}^T\mathbf{X}\,\mathbf{D}\,\mathbf{X}^T\mathbf{p}}. \tag{6.3}$$

The optimal $\mathbf{p}$ is given by solving the minimum eigenvalue problem:

$$\mathbf{X}\,\mathbf{L}\,\mathbf{X}^T\mathbf{p} = \lambda\,\mathbf{X}\,\mathbf{D}\,\mathbf{X}^T\mathbf{p}. \tag{6.4}$$

The eigenvectors $\mathbf{p}_1, \ldots, \mathbf{p}_d$ corresponding to the d smallest eigenvalues are then used as the columns of the projection matrix $\mathbf{P}$, i.e. $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_d]$. The projected samples are obtained by $\mathbf{Y} = \mathbf{P}^T\mathbf{X}$.

6.2.2 Neighborhood Preserving Embedding

Similar to LPP, Neighborhood Preserving Embedding [34] (NPE) tries to preserve the local structure of the data which aims to keep the neighborhood representation in the projection space.

For each sample, local least squares approximation is used to obtain the representative coefficients by the k nearest neighbors. Denote $\mathbf{N}$ the representative matrix which can be obtained by minimizing the following cost function:

$$\phi(\mathbf{N}) = \sum_{\mathbf{x}_j \in \delta_k(\mathbf{x}_i)} \|\mathbf{x}_i - \sum_j N_{ij}\, \mathbf{x}_j\|^2. \tag{6.5}$$

Let $y_i = \mathbf{p}^T \mathbf{x}_i$ be the one dimension projection of $\mathbf{x}_i$. The problem of NPE is given by:

$$\min_{\mathbf{p}} \sum_i (y_i - \sum_j N_{ij}\, y_j)^2. \tag{6.6}$$

With constraint $\mathbf{p}^T \mathbf{X}\, \mathbf{X}^T \mathbf{p} = 1$, Eq.(6.6) becomes:

$$\min_{\mathbf{p}} \frac{\mathbf{p}^T \mathbf{X}\, \mathbf{M}\, \mathbf{X}^T \mathbf{p}}{\mathbf{p}^T \mathbf{X}\, \mathbf{X}^T \mathbf{p}}, \tag{6.7}$$

where $\mathbf{M} = \mathbf{I} - \mathbf{N} - \mathbf{N}^T + \mathbf{N}^T \mathbf{N}$ and $\mathbf{I}$ is the identity matrix.

The corresponding minimum eigenvalue problem is given by:

$$\mathbf{X}\, \mathbf{M}\, \mathbf{X}^T \mathbf{p} = \lambda\, \mathbf{X}\, \mathbf{X}^T \mathbf{p}. \tag{6.8}$$

The eigenvectors $\mathbf{p}_1, \ldots, \mathbf{p}_d$ corresponding to the d smallest eigenvalues are the columns of the sought linear transform $\mathbf{P}$, i.e. $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_d]$. The projected samples are obtained by $\mathbf{Y} = \mathbf{P}^T \mathbf{X}$.

6.2.3 Sparsity Preserving Projection

As LPP tries to preserve the neighborhood structure, Sparsity Preserving Projection [97, 65] (SPP) aims to keep the structure over the whole data set by using sparse representation instead of the linear representation of k nearest neighbors to get the weight matrix. For $\mathbf{x}_i$, the representative coefficients of the rest samples are obtained by solving an ℓ_1 problem:

$$\begin{aligned} \min_{\mathbf{s}_i} \quad & \|\mathbf{s}_i\|_1, \\ s.t. \quad & \mathbf{x}_i = \mathbf{X}\, \mathbf{s}_i, \end{aligned} \tag{6.9}$$

where $\mathbf{s}_i = [s_{i1}, \ldots, s_{i(i-1)}, 0, s_{i(i+1)}, \ldots, s_{in}]^T$. The problem of SPP is:

$$\min_{\mathbf{p}} \sum_i (y_i - \sum_j s_{ij} y_j)^2 = \min_{\mathbf{p}} \sum_i (\mathbf{p}^T \mathbf{x}_i - \mathbf{p}^T \mathbf{X}\, \mathbf{s}_i)^2. \tag{6.10}$$

With the constraint $\mathbf{p}^T\mathbf{X}\,\mathbf{X}^T\mathbf{p} = 1$, Eq.(6.10) becomes:

$$\min_{\mathbf{p}} \frac{\mathbf{p}^T\mathbf{X}(\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S})\mathbf{X}^T\mathbf{p}}{\mathbf{p}^T\mathbf{X}\,\mathbf{X}^T\mathbf{p}} = \max_{\mathbf{p}} \frac{\mathbf{p}^T\mathbf{X}\,\tilde{\mathbf{S}}\mathbf{X}^T\mathbf{p}}{\mathbf{p}^T\mathbf{X}\,\mathbf{X}^T\mathbf{p}}, \tag{6.11}$$

where $\tilde{\mathbf{S}} = \mathbf{S}+\mathbf{S}^T-\mathbf{S}^T\mathbf{S}$, and $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]^T$. The corresponding eigenvalue problem is:

$$\mathbf{X}\,\tilde{\mathbf{S}}\,\mathbf{X}^T\mathbf{p} = \lambda\,\mathbf{X}\,\mathbf{X}^T\mathbf{p}. \tag{6.12}$$

The eigenvectors $\mathbf{p}_1, \ldots, \mathbf{p}_d$ corresponding to the d largest eigenvalues are the columns of the sought linear transform, i.e., $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_d]$, and $\mathbf{Y} = \mathbf{P}^T\mathbf{X}$.

6.2.4 Sparsity preserving discriminant analysis

Cai et al. extended LDA to SDA by adding a geometrically-based regularization term in the objective function of LDA. Qiao et al. added the idea of SPP and proposed SPDA [64].

Let $\mathbf{X}_L = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l]$ denote the label data amtrix. LDA can be regarded as a particular case of a graph-based embedding. Let $\mathbf{S}_w \in \Re^{l\times l}$ and $\mathbf{S}_b \in \Re^{l\times l}$ denote two graph similarity matrices, where $S_w(i,j) = 1/n_c$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to class c; $S_w(i,j) = 0$, otherwise, and $S_b(i,j) = 1/l - S_w(i,j)$. Here n_c is the number of the samples which belongs to class c. The corresponding Laplacian matrices of $\mathbf{S}_w$ and $\mathbf{S}_b$ are represented as $\mathbf{L}_w$ and $\mathbf{L}_b$. The intra-class scatter and the inter-class scatter of LDA can be rewritten as:

$$\mathbf{M}_w = \sum_{c=1}^{C}\sum_{i=1}^{n_c}(\mathbf{x}_i - \bar{\mathbf{x}}_c)(\mathbf{x}_i - \bar{\mathbf{x}}_c)^T = \mathbf{X}_L\mathbf{L}_w\mathbf{X}_L^T, \tag{6.13}$$

and

$$\mathbf{M}_b = \sum_{c=1}^{C}\sum_{i=1}^{n_c}(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^T = \mathbf{X}_L\mathbf{L}_b\mathbf{X}_L^T, \tag{6.14}$$

where $\bar{\mathbf{x}}_c$ is the mean of the samples in the c^{th} class and $\bar{\mathbf{x}}$ is the mean of all the labeled samples.

The SPDA method can be solved by optimizing the following objective function:

$$\mathbf{W} = \arg\max_{\mathbf{W}} \frac{\mathrm{Tr}(\mathbf{W}^T\mathbf{X}_L\mathbf{L}_b\mathbf{X}_L^T\mathbf{W})}{\mathrm{Tr}(\mathbf{W}^T(\mathbf{X}_L(\mathbf{L}_w+\mathbf{L}_b)\mathbf{X}_L^T + \alpha\mathbf{X}\mathbf{L}_\mathbf{S}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W})}, \tag{6.15}$$

where α and β are two balance parameters, and the $\mathbf{L}_\mathbf{S}$ matrix is given by $\mathbf{L}_\mathbf{S} = \mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S}$. The term trace $\mathbf{W}^T\mathbf{X}\mathbf{L}_\mathbf{S}\mathbf{X}^T\mathbf{W}$ represent the sparsity preserving term associated with training data.

The trace ratio can be approximated by: $\mathrm{Tr}((\mathbf{W}^T(\mathbf{X}_L(\mathbf{L}_w + \mathbf{L}_b)\mathbf{X}_L^T + \alpha\mathbf{X}\mathbf{L}_\mathbf{S}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{X}_L\mathbf{L}_b\mathbf{X}_L^T\mathbf{W}))$.

Thus, the optimal $\mathbf{W}$ is given by the eigenvectors of $(\mathbf{W}^T(\mathbf{X}_L(\mathbf{L}_w + \mathbf{L}_b)\mathbf{X}_L^T + \alpha\mathbf{X}\mathbf{L}_\mathbf{S}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{X}_L\mathbf{L}_b\mathbf{X}_L^T\mathbf{W})$ associated with the largest eigenvalues.

6.2.5 Semi-supervised Discriminant Embedding

SDE can be seen as the semi-supervised variant of the Local Discriminant Embedding (LDE) method. In order to discover both geometrical and discriminant structure of the data manifold, SDE relies on three graphs: the within-class graph G_w (intrinsic graph), the between-class graph G_b, and the graph defined over the whole set. For each data sample $\mathbf{x}_i$, two subsets, $N_w(\mathbf{x}_i)$ and $N_b(\mathbf{x}_i)$ are computed. $N_w(\mathbf{x}_i)$ contains the neighbors sharing the same label with $\mathbf{x}_i$ while $N_b(\mathbf{x}_i)$ contains the neighbors having different labels.

Each of the graphs mentioned before, G_w and G_b, is characterized by its corresponding similarity (weight) matrix $\mathbf{S}_w$ and $\mathbf{S}_b$, respectively. The entries of these symmetric matrices are defined by:

$$S_w(i,j) = \begin{cases} sim(\mathbf{x}_i, \mathbf{x}_j), \text{ if } \mathbf{x}_i \in N_w(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_w(\mathbf{x}_i), \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{6.16}$$

$$S_b(i,j) = \begin{cases} sim(\mathbf{x}_i, \mathbf{x}_j), \text{ if } \mathbf{x}_i \in N_b(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_b(\mathbf{x}_i), \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{6.17}$$

Let $\mathbf{L}_w$, $\mathbf{L}_b$ and $\mathbf{L}$ denote the Laplacian matrices associated with the graph similarity matrices $\mathbf{S}_w$, $\mathbf{S}_b$ and $\mathbf{S}$ (the kNN graph), respectively. SDE seeks a linear mapping $\mathbf{W}$ by maximizing the following criterion:

$$\mathbf{W} = \arg\max_{\mathbf{W}} \frac{\mathrm{Tr}(\mathbf{W}^T \mathbf{X}_L \mathbf{L}_b \mathbf{X}_L^T \mathbf{W})}{\mathrm{Tr}(\mathbf{W}^T (\mathbf{X}_L \mathbf{L}_w \mathbf{X}_L^T + \alpha \mathbf{X}\mathbf{L}\mathbf{X}^T + \beta \mathbf{I})\mathbf{W})} \tag{6.18}$$

Let $\tilde{\mathbf{L}}_w \in \Re^{n \times n}$ and $\tilde{\mathbf{L}}_b \in \Re^{n \times n}$ denote the augmented Laplacian matrices, namely:

$$\tilde{\mathbf{L}}_w = \begin{pmatrix} \mathbf{L}_w & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{6.19}$$

$$\tilde{\mathbf{L}}_b = \begin{pmatrix} \mathbf{L}_b & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{6.20}$$

The above criterion becomes:

$$\mathbf{W} = \arg\max_{\mathbf{W}} \frac{\mathrm{Tr}(\mathbf{W}^T \mathbf{X} \tilde{\mathbf{L}}_b \mathbf{X}^T \mathbf{W})}{\mathrm{Tr}(\mathbf{W}^T (\mathbf{X} \tilde{\mathbf{L}}_w \mathbf{X}^T + \alpha \mathbf{X}\mathbf{L}\mathbf{X}^T + \beta \mathbf{I})\mathbf{W})}. \tag{6.21}$$

The trace ratio can be approximated by: $\mathrm{Tr}((\mathbf{W}^T(\mathbf{X}\tilde{\mathbf{L}}_w\mathbf{X}^T + \alpha\mathbf{X}\mathbf{L}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{X}\tilde{\mathbf{L}}_b\mathbf{X}^T\mathbf{W}))$.

Thus, $\mathbf{W}$ is given by the eigenvectors of $(\mathbf{W}^T(\mathbf{X}\tilde{\mathbf{L}}_w\mathbf{X}^T + \alpha\mathbf{X}\mathbf{L}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{X}\tilde{\mathbf{L}}_b\mathbf{X}^T\mathbf{W})$ associated with the largest eigenvalues.

6.2.6 Constrained Graph Embedding

Constrained Graph Embedding [35] (CGE) is a semi-supervised non-linear embedding method which uses the label information as additional constraints mapping the samples with a same label to one point in the projection space. We assume that the first l samples are with labels from c classes. In the projection space, a constraint matrix $\mathbf{U}$ is used to keep the samples with a same label in one point. The definition of $\mathbf{U}$ is as follows:

$$\mathbf{U} = \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{6.22}$$

where $\mathbf{U} \in \Re^{n \times (c+(n-l))}$, $\mathbf{I}$ is the $(n-l) \times (n-l)$ identity matrix and the i-th row of $\mathbf{J}$ is an indictor vector of $\mathbf{x}_i$:

$$J_{ij} = \begin{cases} 1, \text{ if } \mathbf{x}_i \text{ is labeled from class } j, \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{6.23}$$

where $j = 1, \ldots, c$.

An auxiliary vector $\mathbf{z}$ is adopted to implement the constraint ($\mathbf{y}$ is the one-dimensional map of data matrix $\mathbf{X}$):

$$\mathbf{y} = \mathbf{U}\,\mathbf{z}. \tag{6.24}$$

With the above constraint, it is clear to see that if $\mathbf{x}_i$ and $\mathbf{x}_j$ share the same label, then $y_i = y_j$.

With simple algebraic formulation, we have:

$$\sum_{i,j} (y_i - y_j)^2 W_{ij} = \mathbf{y}^T \mathbf{L}\,\mathbf{y} = \mathbf{z}^T \mathbf{U}^T \mathbf{L}\,\mathbf{U}\,\mathbf{z}, \tag{6.25}$$

and

$$\mathbf{y}^T \mathbf{D}\,\mathbf{y} = \mathbf{z}^T \mathbf{U}^T \mathbf{D}\,\mathbf{U}\,\mathbf{z}, \tag{6.26}$$

where the affinity matrix $\mathbf{W}$ can be given by the simple kNN graph as defined in Eq.(3.1), $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{D}$ is diagonal with $D_{ii} = \sum_j W_{ij}$.

The problem of CGE is as follows:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{z}^T \mathbf{U}^T \mathbf{L}\,\mathbf{U}\,\mathbf{z}, \\ s.t. \quad & \mathbf{z}^T \mathbf{U}^T \mathbf{D}\,\mathbf{U}\,\mathbf{z} = 1. \end{aligned} \tag{6.27}$$

The optimal vector $\mathbf{z}$ is given by the minimum eigenvalue solution to the generalized eigenvalue problem:

$$\mathbf{U}^T \mathbf{L}\,\mathbf{U}\,\mathbf{z} = \lambda\,\mathbf{U}^T \mathbf{D}\,\mathbf{U}\,\mathbf{z}. \tag{6.28}$$

The eigenvectors $\mathbf{z}_1, \ldots, \mathbf{z}_d$ corresponding to the d smallest eigenvalues, yield the auxiliary matrix $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_d]$. We have $\mathbf{Y} = (\mathbf{U}\mathbf{Z})^T$. $\mathbf{Y}$ is a $d \times n$ matrix and it represents the data projection of $\mathbf{X}$.

6.2.7 Flexible Manifold Embedding

Flexible Manifold Embedding [57] (FME) is a label propagation method. FME can simultaneously estimate a prediction matrix over the whole input samples and an approximate linear projection from the original samples to their prediction vectors by a regression term.

FME algorithm solves the following problem:

$$\begin{aligned}\min_{\mathbf{F},\mathbf{P},\mathbf{b}} \operatorname{Tr}(\mathbf{F}-\mathbf{T})^T \Lambda\,(\mathbf{F}-\mathbf{T}) + \operatorname{Tr}(\mathbf{F}^T\mathbf{L}\,\mathbf{F}) \\ +\mu\,(\|\mathbf{P}\|^2 + \gamma\|\mathbf{X}^T\mathbf{P} + \mathbf{1}\,\mathbf{b}^T - \mathbf{F}\|^2)\end{aligned} \tag{6.29}$$

where Λ is a diagonal matrix with the first l and the rest u diagonal elements as 1 and 0. $\mathbf{L}$ is a graph Laplacian matrix of some similarity matrix. $\mathbf{P}$ is the unknown projection matrix and $\mathbf{b}$ is a bias vector. Assuming c is the number of classes, $\mathbf{T}$ in Eq.(6.29) is a $n \times c$ matrix: $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n]^T \in \Re^{n\times c}$, where $t_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j, 0 otherwise. The task is to estimate the label indicator matrix $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]^T \in \Re^{n\times c}$ so that the label of unlabeled samples can be inferred.

The solution to Eq.(6.29) is as follows:

$$\mathbf{F} = (\Lambda + \mathbf{L} + \mu\gamma\,\mathbf{H}_c - \mu\gamma^2\,\mathbf{N})^{-1}\Lambda\,\mathbf{T}, \tag{6.30}$$

where $\mathbf{N} = \mathbf{X}_c^T(\gamma\,\mathbf{X}_c\,\mathbf{X}_c^T + \mathbf{I})^{-1}\mathbf{X}_c$ and $\mathbf{X}_c = \mathbf{X}\,\mathbf{H}_c$ with $\mathbf{H}_c = \mathbf{I} - \frac{1}{n}\mathbf{1}\,\mathbf{1}^T$. $\mathbf{I}$ is an identity matrix of size n.

6.3 Conclusion

In this chapter, we reviewed several manifold embedding methods including unsupervised and semi-supervised algorithms. We presented in some details three unsupervised embedding algorithms (namely LPP, NPE and SPP) and four semi-supervised embedding techniques (namely SPDA, SDE, CGE and FME).

7

Flexible Constrained Sparsity Preserving Embedding

Abstract

In this chapter, we propose a semi-supervised learning method named Constrained Sparsity Preserving Embedding (CSPE). Afterwards, we introduce another flexible semi-supervised embedding method named Flexible Constrained Sparsity Preserving Embedding (FCSPE).

The framework of CSPE does not provide a straightforward solution to the out-of-sample problem. Indeed, the regression is carried out as an extra step. With the flexible method, FCSPE, both the non-linear mapping and the regression are simultaneously estimated within one single framework. This will lead to more flexibility in the final solution as shown by the experimental results.

Contents

7.1 Introduction

SPP is a successful unsupervised learning method. To extend SPP to a semi-supervised embedding method, we introduce the idea of in-class constraints in CGE into SPP and propose a new semi-supervised method for data embedding named Constrained Sparsity Preserving Embedding (CSPE). The weakness of CSPE is that it can not handle the new coming samples which means a cascade

regression should be performed after the non-linear mapping is obtained by CSPE over the whole training samples. Inspired by FME, we add a regression term in the objective function to obtain an approximate linear projection simultaneously when non-linear embedding is estimated and proposed Flexible Constrained Sparsity Preserving Embedding (FCSPE). So in this chapter, two semi-supervised embedding methods namely CSPE and FCSPE are proposed. Compared to the existing works, the proposed CSPE retains the advantages of both CGE and SPP. On the other hand, the proposed FCSPE simultaneous estimates the non-linear mapping over the training samples and the linear projection for solving the out-of-sample problem, which is usually not provided by existing graph-based semi-supervised non-linear mapping methods.

7.2 Constrained Sparsity Preserving Embedding (CSPE)

In this section, we introduce a semi-supervised embedding method named Constrained Sparsity Preserving Embedding (CSPE). In CSPE, the construction of the affinity matrix is parameter free and the sparse structure is preserved in the projection space. In addition, the idea of in-class constraints from CGE is also integrated in the algorithm which merges the sample points with the same label together in the projection space. The affinity matrix is obtained by sparse representation as in Eq.(6.9). In this way, the affinity matrix can be obtained without setting any parameters.

To keep the sparse representation in the projection space as explained in SPP, every y_i should have a similar combination of the rest samples in the projection space as x_i in the original space. So the problem of CSPE is the same like in SPP:

$$\min_{\mathbf{y}} \sum_i (y_i - \mathbf{y}^T \mathbf{s}_i)^2, \tag{7.1}$$

where $\mathbf{s}_i$ is provided by Eq.(6.9). $\mathbf{y}$ is the 1D projection of the whole set. The definition of the constraint matrix $\mathbf{U}$ is the same as Eq.(6.22) to constrain the projection sample points, i.e. let $\mathbf{y} = \mathbf{U}\,\mathbf{z}$. We note $y_i = \mathbf{y}_*^T \mathbf{e}_i$, where the i-th item of the vector $\mathbf{e}_i$ is 1, 0 otherwise. Then, Eq.(7.1) becomes:

$$\min_{\mathbf{z}} \sum_i (\mathbf{z}^T \mathbf{U}^T \mathbf{e}_i - \mathbf{z}^T \mathbf{U}^T \mathbf{s}_i)^2. \tag{7.2}$$

With simple algebraic manipulations, the objective function can be rewritten as:

$$\begin{aligned} \sum_i (\mathbf{z}^T \mathbf{U}^T \mathbf{e}_i - \mathbf{z}^T \mathbf{U}^T \mathbf{s}_i)^2 &= \mathbf{z}^T \mathbf{U}^T (\mathbf{I} - \mathbf{S} - \mathbf{S}^T + \mathbf{S}^T \mathbf{S}) \mathbf{U}\,\mathbf{z} \\ &= \mathbf{z}^T \mathbf{U}^T \mathbf{U}\,\mathbf{z} - \mathbf{z}^T \mathbf{U}^T \tilde{\mathbf{S}}\,\mathbf{U}\,\mathbf{z}, \end{aligned} \tag{7.3}$$

where $\tilde{\mathbf{S}} = \mathbf{S} + \mathbf{S}^T - \mathbf{S}^T \mathbf{S}$. With the constraint $\mathbf{z}^T \mathbf{U}^T \mathbf{U}\,\mathbf{z} = 1$, the objective function Eq.(7.3) can be recasted into:

$$\max_{\mathbf{z}} \frac{\mathbf{z}^T \mathbf{U}^T \tilde{\mathbf{S}} \mathbf{U} \mathbf{z}}{\mathbf{z}^T \mathbf{U}^T \mathbf{U} \mathbf{z}}. \tag{7.4}$$

The optimal vector $\mathbf{z}$ is given by the maximum eigenvalue solution to the following generalized eigenvalue problem:

$$\mathbf{U}^T \tilde{\mathbf{S}} \mathbf{U} \mathbf{z} = \lambda \mathbf{U}^T \mathbf{U} \mathbf{z}. \tag{7.5}$$

The eigenvectors $\mathbf{z}_1, \ldots, \mathbf{z}_d$ corresponding to the d largest eigenvalues, yield the auxiliary matrix $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_d]$. The data projections of $\mathbf{X}$ in the d-dimensional space is given by:

$$\mathbf{Y} = (\mathbf{U} \mathbf{Z})^T. \tag{7.6}$$

The projection matrix of CSPE could not be obtained directly since C-SPE provides a non-linear mapping. The traditional way to deal with a new incoming sample is to re-perform the whole algorithm again which will be time-consuming.

We assume a linear projection $\mathbf{y}(\mathbf{x}) = \mathbf{P}^T \mathbf{x}$ to approximate the original non-linear mapping, where $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_d]$. One simple idea to calculate the matrix $\mathbf{P}$ is to fit the following function which minimizes the least square error on the existing samples.

$$\mathbf{P} = \arg\min_{\mathbf{P}} \Big(\|\mathbf{P}^T \mathbf{X} - \mathbf{Y}\|^2 + \gamma \|\mathbf{P}\|^2 \Big), \tag{7.7}$$

where γ is a positive balance parameter that controls the regularization.

By vanishing the derivative of the right side w.r.t. $\mathbf{P}$, the optimal $\mathbf{P}$ can be obtained as:

$$\mathbf{P} = (\mathbf{X}\mathbf{X}^T + \gamma \mathbf{I})^{-1} \mathbf{X}\mathbf{Y}^T. \tag{7.8}$$

For a new incoming sample $\mathbf{x}_{test}$, its embedding $\mathbf{y}_{test}$ is given by

$$\mathbf{y}_{test} = \mathbf{P}^T \mathbf{x}_{test} = ((\mathbf{X}\mathbf{X}^T + \gamma \mathbf{I})^{-1} \mathbf{X}\mathbf{U}\mathbf{Z})^T \mathbf{x}_{test}. \tag{7.9}$$

Over the training samples, the approximate linear projection $\mathbf{P}$ can also be used.

7.3 Flexible Constrained Sparsity Preserving Embedding (FCSPE)

CSPE is a semi-supervised embedding method which does not provide a direct linear projection within its criterion. Thus, in order to deal with unseen data samples, regression should be applied to estimate a linear projection as a cascade process after the non-linear embedding is obtained. To deal with the out-of-sample problem, we introduce Flexible Constrained Sparsity P-reserving Embedding (FCSPE), a new flexible embedding method based on

CSPE which simultaneously utilizes the core idea of CSPE and generates a projection matrix by optimizing one objective criterion. This criterion aims at getting both the non-linear representations and the linear regression based on minimizing the residual errors (choosing a linear subspace that are close to the non-linear one). This flexible optimization is usually better then a cascade estimation which simultaneously computes non-linear embedding of the samples and the regression over these non-linear representations.

Assuming $\mathbf{Y}^T = \mathbf{X}^T\mathbf{P} + \mathbf{1}\,\mathbf{b}^T$, we try to minimize the following optimal function:

$$\begin{aligned} \min_{\mathbf{Z},\mathbf{P},\mathbf{b}} \quad & \mathrm{Tr}(\mathbf{Z}^T\mathbf{U}^T(\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S})\mathbf{U}\,\mathbf{Z}) \\ & + \mu(\|\mathbf{P}\|^2 + \gamma\,\|\mathbf{X}^T\mathbf{P} + \mathbf{1}\,\mathbf{b}^T - \mathbf{U}\,\mathbf{Z}\|^2), \\ s.t. \quad & \mathbf{Z}^T\mathbf{U}^T\mathbf{U}\,\mathbf{Z} = \mathbf{I}. \end{aligned} \tag{7.10}$$

Define cost function $Q(\mathbf{Z},\mathbf{P},\mathbf{b})$ as follows:

$$\begin{aligned} Q(\mathbf{Z},\mathbf{P},\mathbf{b}) = & \mathrm{Tr}(\mathbf{Z}^T\mathbf{U}^T(\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S})\mathbf{U}\,\mathbf{Z}) \\ & + \mu(\|\mathbf{P}\|^2 + \gamma\,\|\mathbf{X}^T\mathbf{P} + \mathbf{1}\,\mathbf{b}^T - \mathbf{U}\,\mathbf{Z}\|^2). \end{aligned} \tag{7.11}$$

Let $\partial Q(\mathbf{Z},\mathbf{P},\mathbf{b})/\partial\mathbf{b} = 0$, we obtain

$$\mathbf{b} = \frac{1}{n}\mathbf{Z}^T\mathbf{U}^T\mathbf{1} - \mathbf{P}^T\mathbf{X}\,\mathbf{1}. \tag{7.12}$$

Then we can have $Q(\mathbf{Z},\mathbf{P})$.

Again let $\partial Q(\mathbf{Z},\mathbf{P})/\partial\mathbf{P} = 0$, we get

$$\mathbf{P} = \gamma\,(\gamma\,\mathbf{X}\,\mathbf{H}_c\mathbf{X}^T + \mathbf{I})^{-1}\mathbf{X}\,\mathbf{H}_c\mathbf{U}\,\mathbf{Z} = \mathbf{A}\,\mathbf{U}\,\mathbf{Z}, \tag{7.13}$$

where $\mathbf{H}_c = \mathbf{I} - \frac{1}{n}\mathbf{1}\,\mathbf{1}^T$ and $\mathbf{A} = \gamma\,(\gamma\,\mathbf{X}\,\mathbf{H}_c\mathbf{X}^T + \mathbf{I})^{-1}\mathbf{X}\,\mathbf{H}_c$.

We have $\mathbf{X}^T\mathbf{P} + \mathbf{1}\,\mathbf{b}^T = \mathbf{H}_c\mathbf{X}^T\mathbf{A}\,\mathbf{U}\,\mathbf{Z} + \frac{1}{n}\mathbf{1}\,\mathbf{1}^T\mathbf{U}\,\mathbf{Z} = \mathbf{B}\,\mathbf{U}\,\mathbf{Z}$, where $\mathbf{B} = \mathbf{X}\,\mathbf{H}_c\mathbf{A} + \frac{1}{n}\mathbf{1}\,\mathbf{1}^T$. Hence,

$$Q(\mathbf{Z}) = \mathrm{Tr}(\mathbf{Z}^T\mathbf{E}\,\mathbf{Z}), \tag{7.14}$$

where $\mathbf{E} = \mathbf{U}^T(\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S})\,\mathbf{U} + \mu\,\mathbf{U}^T\mathbf{A}^T\mathbf{A}\,\mathbf{U} + \mu\gamma\,\mathbf{U}\,(\mathbf{B}-\mathbf{I})^T(\mathbf{B}-\mathbf{I})\,\mathbf{U}$.

Now we want to solve the following problem which is only related to $\mathbf{Z}$:

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \mathrm{Tr}(\mathbf{Z}^T\mathbf{E}\,\mathbf{Z}), \\ s.t. \quad & \mathbf{Z}^T\mathbf{U}^T\mathbf{U}\,\mathbf{Z} = \mathbf{I}. \end{aligned} \tag{7.15}$$

Then a Lagrangian multiplier is used. The optimal vector $\mathbf{z}$ is given by the minimum eigenvalue solution to the following generalized eigenvalue problem:

$$\mathbf{E}\,\mathbf{z} = \lambda\,\mathbf{U}^T\mathbf{U}\,\mathbf{z}. \tag{7.16}$$

The eigenvectors $\mathbf{z}_1,\ldots,\mathbf{z}_d$ corresponding to the d smallest eigenvalues, yield the auxiliary matrix $\mathbf{Z} = [\mathbf{z}_1,\ldots,\mathbf{z}_d]$.

Then the approximate linear projection matrix $\mathbf{P}$ is given by Eq.(7.13), i.e. $\mathbf{P} = \mathbf{A}\,\mathbf{U}\,\mathbf{Z}$.

The linear projection can be used over both the training samples and the testing samples.

7.4 Performance evaluation

In this section, we evaluate the proposed methods on eight real image databases: Yale, ORL, FERET, PIE, Extended Yale B, LFW (the original data set and the aligned version), COIL-20, and USPS.

7.4.1 Comparisons of effectiveness

To evaluate the proposed methods, we compare them with several competing methods including Locality Preserving Projection (LPP) [36], Sparsity Preserving Projection (SPP) [65], Sparsity preserving discriminant analysis (SPDA) [64], Semi-supervised Discriminant Embedding (SDE) [101], and Constrained Graph Embedding (CGE) [35]. We also use three label propagation methods Gaussian Random Fields (GRF) [114], Robust multi-class Graph Transduction (RMGT) [51], and FME [57] which can be regarded as classifiers.

The FCSPE method has two parameters to tune: μ and γ. In the experiments, they are chosen from $\{10^{-6}, 10^{-3}, 1, 10^{3}, 10^{6}\}$. For every method, several values for the parameters are used. We then report the top-1 recognition accuracy (best average recognition rate) of all methods from the best parameter configuration. In all experiments, PCA is used as a preprocessing step to preserve 98% energy of the data.

For every data sets except LFW-a, we randomly choose 50% of samples to be in the training set and the rest 50% form the test set. For each class, l samples are randomly chosen (from the training set) as labeled samples ($l = 1, 2$ & 3). All compared methods use the training set (labeled and unlabeled samples) to build the projection model. Then, the obtained projection of the unlabeled train data and test data samples are classified using the Nearest Neighbor (NN) classifier. We repeat every experiments 10 times, i.e., we generate 10 random splits (Train/Test) for every data set. We depict the average recognition rates over the 10 splits. In general, the recognition rates varies with the dimension of the subspace. Thus, the average recognition rate is given by a curve depicting the rate as a function of the dimension of the new subspace.

Tables 7.1-7.7 show the average recognition rates and the standard deviations for six databases. For each database, 1-3 samples for each class are labeled and recognition rates are shown on both unlabeled training sets and testing sets. In the tables, 'Unlabeled' means the unlabeled training set and 'Test' means the testing set.

We conducted one experiment on LFW and LFW-a. The main purpose of this data set is to evaluate face verification in the wild. Since we are addressing face recognition problem (one to many matching), we use another protocol for evaluating the proposed methods. For every person we randomly select 7 images and the remaining 4 images are used for testing. These test images are used as unlabeled data sample. In other worlds, the whole data set is used for training. We just test on one split using the original color feature and the LBP image [60, 79], respectively. For comparison, we use SPDA, SDE, FME and CGE. The NN classifier is used for classification. Table 7.8 shows the recognition rates for LFW-a and Table 7.9 shows the recognition rates for LFW.

Table 7.1: Average recognition rates (%) on Yale.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF [114]	66.5±12.0	-	70.8±12.1	-	70.4±12.5	-
RMGT [51]	69.9±10.4	-	72.3±11.1	-	71.1±12.0	-
LPP [36]	79.2±7.4	82.5±6.8	81.5±5.9	86.9±3.8	83.6±6.4	87.9±4.2
SPP [65]	80.8±5.3	83.6±8.1	86.0±5.8	88.9±4.6	86.7±4.2	91.5±5.4
SPDA [64]	76.5±7.1	80.7±8.1	83.0±7.7	88.6±3.1	84.4±7.8	91.4±2.9
SDE [101]	63.6±12.0	72.2±13.4	71.5±8.8	80.4±6.1	73.4±9.8	86.1±5.5
FME [57]	74.1±8.8	75.6±8.3	78.8±9.4	82.6±4.8	79.8±9.9	85.4±4.4
CGE [35]	77.1±9.0	73.8±6.9	82.5±7.8	82.1±6.9	82.9±12.8	86.3±7.1
CSPE	83.1±4.8	83.5±6.7	90.5±5.0	89.4±3.0	94.9±3.6	**93.5±5.2**
FCSPE	**83.6±4.3**	**82.9±7.6**	**90.3±5.3**	**89.6±3.4**	**95.1±3.3**	92.4±5.2

Table 7.2: Average recognition rates (%) on ORL.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	59.1±6.1	-	68.7±5.1	-	75.4±4.8	-
RMGT	60.1±5.9	-	69.9±4.9	-	76.9±4.4	-
LPP	65.5±4.0	61.1±4.1	73.1±5.0	69.1±4.5	77.4±4.3	75.3±3.0
SPP	67.9±2.8	63.6±4.8	79.0±4.4	76.4±5.5	87.6±3.1	84.0±2.1
SPDA	56.8±4.0	56.4±4.9	72.3±2.8	74.4±5.2	83.5±3.3	84.8±1.7
SDE	38.9±2.4	44.9±6.1	50.1±5.2	60.4±4.1	62.0±5.4	72.6±3.4
FME	58.6±5.2	56.0±5.0	76.0±4.3	75.3±4.8	84.8±3.7	84.6±1.9
CGE	61.3±6.5	56.4±3.6	74.7±7.0	70.8±5.3	83.5±5.5	81.8±5.4
CSPE	66.9±3.7	62.0±4.3	79.8±5.7	78.9±3.7	88.4±2.8	88.5±1.8
FCSPE	**68.5±3.1**	**64.5±4.1**	**82.8±3.0**	**80.0±3.7**	**88.8±2.4**	**88.8±2.0**

Table 7.3: Average recognition rates (%) on FERET.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	17.8±10.8	-	24.5±15.3	-	31.1±22.4	-
RMGT	18.6±10.6	-	24.9±15.5	-	32.0±23.3	-
LPP	17.3±11.7	18.3±8.8	21.3±14.4	26.0±9.0	27.2±22.6	31.5±6.5
SPP	37.7±16.9	29.0±15.0	50.4±20.0	42.0±13.9	57.3±30.5	53.4±12.2
SPDA	24.3±15.6	23.6±12.9	45.5±21.0	45.9±16.8	53.3±33.0	65.7±17.0
SDE	20.6±13.2	20.0±11.8	30.8±15.5	33.6±13.4	39.2±27.3	47.6±12.3
FME	25.3±17.5	24.5±12.3	44.2±22.9	46.6±13.6	57.3±34.9	64.0±13.3
CGE	24.7±11.5	24.3±12.5	40.1±14.4	40.3±15.1	55.1±26.5	57.6±13.2
CSPE	39.6±15.6	34.4±15.5	58.3±20.2	55.7±17.1	67.5±31.2	69.0±13.7
FCSPE	**40.8±13.0**	**37.3±13.2**	**60.7±18.3**	**62.2±15.9**	**70.4±28.2**	**73.3±15.2**

Table 7.4: Average recognition rates (%) on PIE.

Labeled	1		2		3	
Methods	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	9.8±5.8	-	16.9±6.8	-	22.2±5.9	-
RMGT	10.3±5.7	-	17.3±6.7	-	22.5±5.9	-
LPP	11.0±5.2	11.3±2.7	16.4±7.4	17.0±4.0	21.9±6.8	18.5±4.5
SPP	20.7±6.8	25.0±5.6	32.0±7.2	36.0±5.1	38.7±6.6	41.9±7.0
SPDA	14.3±6.0	17.9±3.7	29.4±10.7	35.1±9.7	41.8±7.0	46.8±9.6
SDE	15.2±5.5	19.8±5.0	24.5±6.6	29.3±5.6	31.8±5.7	35.3±6.8
FME	9.6±4.7	10.5±2.7	20.5±9.4	23.2±6.9	29.6±7.6	29.5±7.6
CGE	18.7±6.3	22.5±5.5	29.7±6.4	34.0±5.8	37.1±5.2	39.9±7.4
CSPE	27.6±9.4	**31.0±7.5**	36.9±8.6	41.1±7.6	46.7±6.1	50.4±8.3
FCSPE	**27.7±9.3**	31.0±4.3	**39.3±7.9**	**43.0±6.4**	**48.2±7.7**	**51.9±8.1**

Table 7.5: Average recognition rates (%) on Extended Yale B.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	25.0±19.8	-	41.7±16.8	-	50.1±8.8	-
RMGT	30.0±19.2	-	44.2±16.3	-	53.4±6.4	-
LPP	25.8±17.2	24.1±13.1	40.2±15.6	38.6±14.3	47.6±8.7	43.8±8.9
SPP	36.8±17.4	34.7±14.3	52.4±18.4	51.6±19.2	63.4±8.2	60.0±10.1
SPDA	36.5±17.3	34.7±13.9	52.8±18.6	50.9±18.3	61.8±12.0	57.7±11.9
SDE	33.6±15.2	32.4±13.3	50.9±18.6	50.4±18.4	61.9±9.2	59.1±9.9
FME	28.8±20.2	27.1±16.5	47.7±20.4	46.3±19.8	58.3±12.0	54.5±11.7
CGE	36.6±14.9	34.9±12.7	51.8±16.3	49.7±15.9	62.5±9.6	58.4±10.1
CSPE	**41.6±17.0**	39.1±15.1	57.4±18.1	56.0±18.6	68.4±8.4	64.5±9.9
FCSPE	**41.6±16.7**	**39.3±15.2**	**57.5±18.1**	**55.8±18.7**	**68.8±8.8**	**64.7±9.9**

Table 7.6: Average recognition rates (%) on USPS.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	33.4±8.6	-	46.8±6.9	-	56.2±5.7	-
RMGT	48.5±5.5	-	55.6±6.6	-	61.8±5.3	-
LPP	43.4±7.0	43.3±5.9	54.3±5.9	52.8±5.6	61.6±4.5	59.8±3.7
SPP	40.8±5.0	40.7±4.5	52.3±4.4	50.6±4.4	59.3±4.3	57.3±3.6
SPDA	33.0±5.4	33.1±6.0	43.5±3.7	42.8±4.5	51.3±3.1	51.1±3.3
SDE	29.0±6.7	29.3±5.8	35.6±5.6	35.7±4.5	40.6±3.6	40.3±4.1
FME	31.4±6.0	31.3±6.5	45.3±4.6	44.4±4.9	53.9±4.1	52.5±3.0
CGE	37.9±9.0	38.3±4.3	49.3±5.5	49.7±4.4	57.6±4.9	56.1±3.3
CSPE	43.0±6.6	42.6±6.6	56.9±5.7	**55.1±5.0**	64.4±3.1	**62.4±3.7**
FCSPE	**42.9±7.2**	**41.5±6.7**	**57.8±5.**4	55.0±5.1	**65.5±3.8**	62.2±3.5

Table 7.7: Average recognition rates (%) on COIL-20.

Labeled	1		2		3	
Method	Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
GRF	58.4±4.8	-	64.4±4.6	-	68.3±4.4	-
RMGT	60.7±4.6	-	66.1±4.2	-	72.3±4.5	-
LPP	61.9±4.7	56.7±3.8	68.6±3.8	63.4±4.9	72.7±5.0	67.1±3.7
SPP	64.8±6.8	60.2±4.4	72.2±3.8	66.5±5.1	76.1±6.3	71.2±5.2
SPDA	56.4±4.7	53.7±4.4	64.1±3.3	63.6±4.9	68.8±3.9	69.2±5.0
SDE	42.2±4.5	41.8±3.0	46.6±3.4	50.6±3.8	53.4±5.1	58.2±5.6
FME	57.9±5.2	54.6±3.4	66.4±2.9	64.8±5.4	71.0±4.5	71.7±5.5
CGE	63.6±8.8	60.2±6.3	71.3±5.1	69.8±8.8	75.5±4.9	74.3±6.1
CSPE	**67.1±6.8**	**63.7±5.0**	70.6±5.1	69.7±5.4	76.3±5.3	73.6±5.7
FCSPE	65.8±6.9	60.9±5.6	**73.6±6.5**	**70.9±8.5**	**77.7±6.5**	**73.3±6.0**

Table 7.8: Recognition rates (%) on LFW-a.

Method \ Feature	Raw images	LBP feature
SPDA	43.6	23.2
SDE	20.9	18.4
FME	35.3	22.7
CGE	30.9	41.0
CSPE	41.3	45.9
FCSPE	**45.0**	**48.9**

Table 7.9: Recognition rates (%) on LFW.

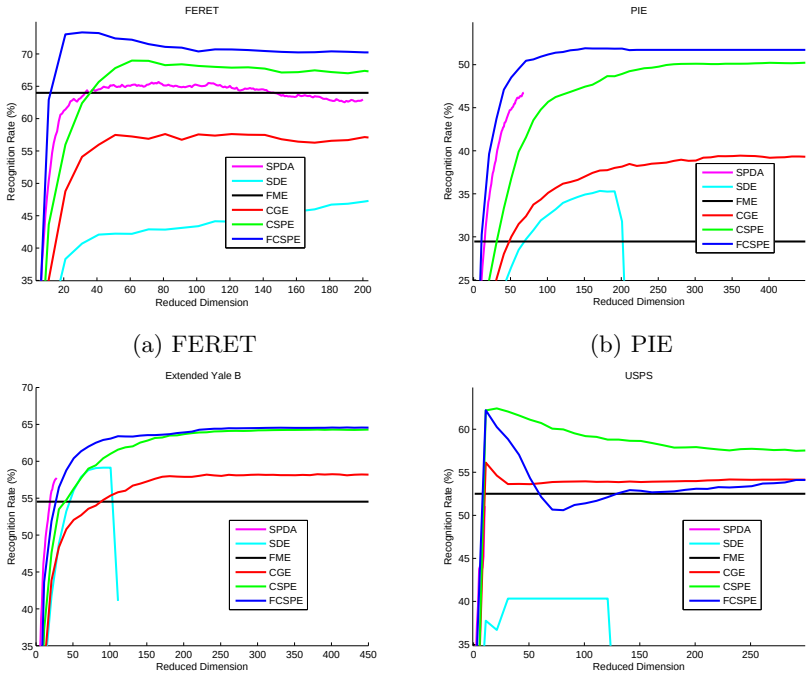
Method \ Feature	Raw images	LBP feature
SPDA	15.1	12.1
SDE	16.0	14.5
FME	23.0	12.8
CGE	23.9	29.6
CSPE	30.9	33.0
FCSPE	**33.0**	**33.2**

Figure 7.1 illustrates the average recognition rate curves among the range of feature dimension. NN classifier is used for classification. These curves are obtained on the test set with three labeled samples per class. We recall that FME method does not depend on the dimension since it is a label propagation technique. We stress the fact that the range of feature dimensions is not the same for all compared methods. Thus, the maximum dimensions of the methods are not the same. The maximum dimension of SPDA method is given by $c-1$, and that of SDE is given by the dimension of input samples. For CGE, CSPE and FCSPE, the maximum dimension relying on the constraint matrix $\mathbf{U}$ is given by $n-l+c$.

Figure 7.2 illustrates the average recognition rate curves among the range of reduced dimension as well. The classifier is non-linear SVM, and the rest settings are the same as those in Figure 7.1.

Analysis of results: According to the results reported in the previous tables and figures, we can make the following observations:

- The proposed CSPE and FCSPE algorithms can out-perform the other embedding algorithms both on the unlabeled training sets and the test sets. These results show the the effectiveness of CSPE and FCSPE.
- Almost all algorithms perform better on the unlabeled training sets than on the test sets for most of the datasets. This is intuitive since the unlabeled training sets are used in the learning model.
- The CSPE and FCSPE algorithms can outperform the other algorithms for NN and SVM classifier. According to the figures, the superiority of the proposed methods are independent from the classifier used.
- In most cases, the proposed FCSPE algorithm provides better performance than the proposed CSPE algorithm. As we mentioned in the previous section, FCSPE simultaneously computes a non-linear embedding of training samples and the linear transform based on a regression over these non-linear representations. This provides better data projections than those obtained by CSPE which performs a cascaded estimation, in the sense it calculates the non-linear embedding first and then perform a linear regression over the non-linear embedding.



(a) FERET (b) PIE

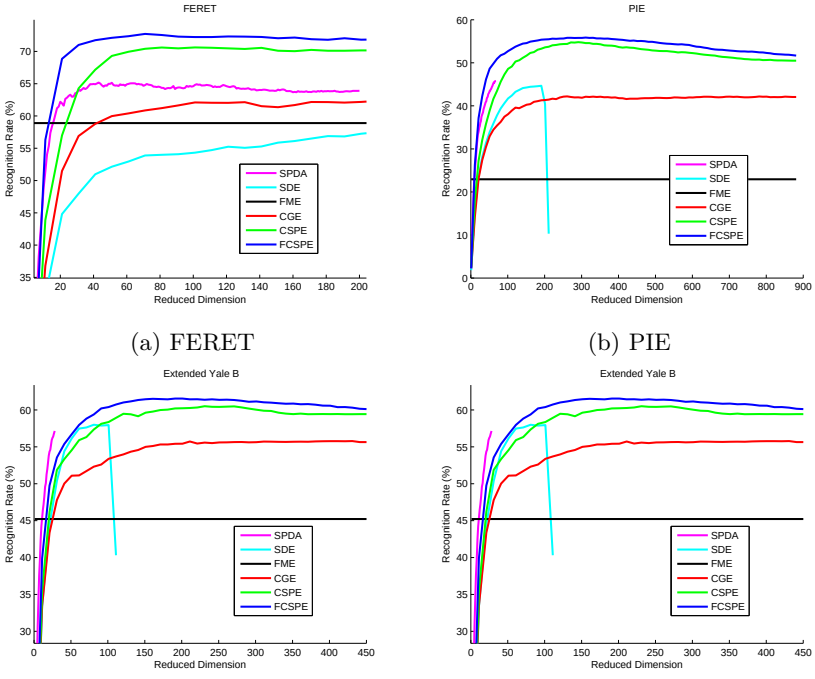
(c) Extended Yale B (d) USPS

Fig. 7.1: Recognition rates of different embedding methods as a function of feature dimension for FERET, PIE, Extended Yale B and USPS data sets (test evaluation). Three samples per class are labeled. The classifier is NN.

- According to the depicted curves, we can observe that the two proposed methods provide good performance even with small feature dimensions. This means that the proposed method can out-perform the other methods even in the low dimensional projection spaces.
- For the face images captured in the wild, both proposed methods keep their superiority with respect to the competing methods even for the misaligned faces. Moreover, the results show that the out-performance of the proposed methods can also be obtained with other types of image descriptors.
- In most cases, the performance of FCSPE is superior to the CSPE algorithm even for the unlabeled train set.

7.4.2 Sensitivity to parameters

The FCSPE method has two parameters μ and γ. In this section, we aim to study the recognition rates obtained by FCSPE when these two parameters vary. Figure 7.3 illustrates the recognition rates with different parameter



(a) FERET (b) PIE

(c) Extended Yale B (d) USPS

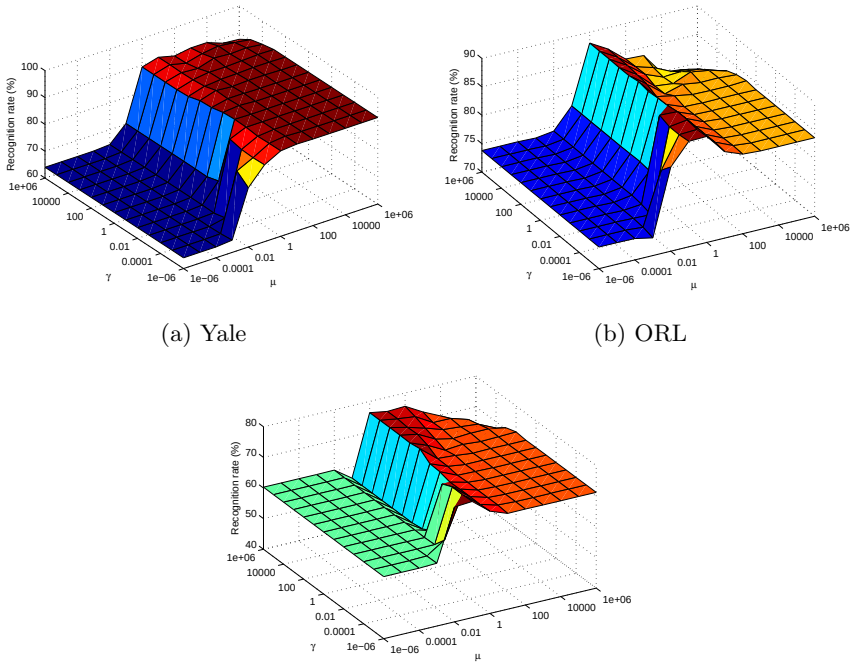
Fig. 7.2: Recognition rates of different embedding methods as a function of feature dimension for FERET, PIE, Extended Yale B and USPS data sets (test evaluation). Three samples per class are labeled. The classifier is SVM.

values for Yale, ORL and COIL-20 datasets. Figures 7.4 illustrates the recognition rates with a smaller range of parameter μ (from 10^{-2} to 10^{2}) for Yale, ORL and COIL-20.

Table 7.10: The comparison between the recognition rates (%) of the optimal parameters and the fixed parameters($\mu = 1, \gamma$=1)

Parameters \ Datasets	Yale	ORL	COIL-20
Optimal	92.2	88.8	76.3
Fixed	91.1	88.1	74.3

We can observe that the optimal domain for the two parameters μ and γ is almost the same for the three face datasets. Generally, μ should be near to 1 and the influence of γ is not significant when μ is fixed. We can conclude that



(a) Yale (b) ORL

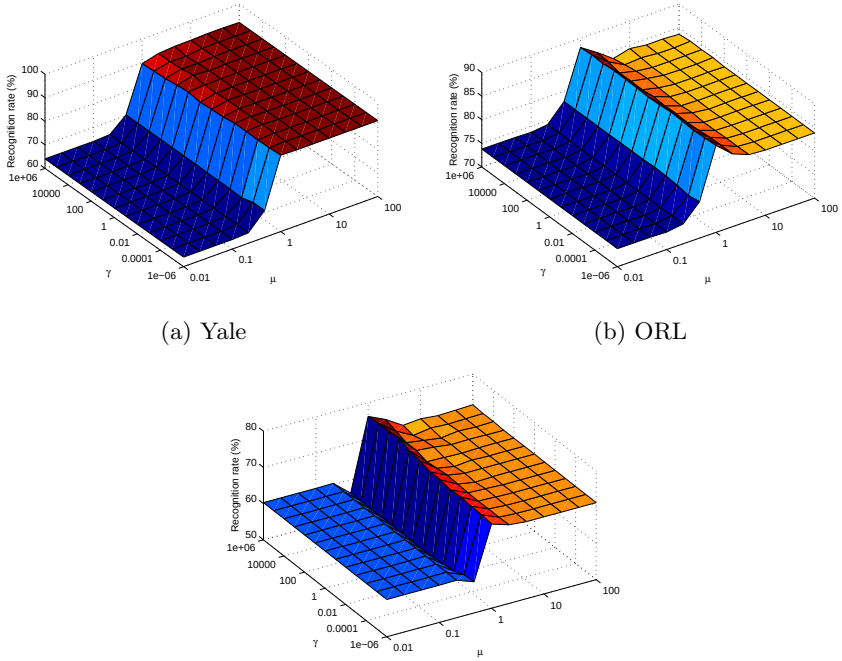
(c) COIL-20

Fig. 7.3: Recognition rates variation as a function of different values of parameter μ and γ on Yale, ORL and COIL-20.

despite the fact that our proposed algorithms have two parameters, optimal values of theses parameters are simply limited to a small interval of values. The setting of these parameters is not a difficult task. Table 7.10 depicts a comparison between the recognition rates obtained with fixed parameters ($\mu = 1, \gamma = 1$) and the rates obtained with the optimal ones. This shows that by fixing the two parameters to the values of $\mu = 1, \gamma = 1$ the associated performance is almost similar to the optimal results. In this case, one can simply fix the parameters when the proposed method is used.

7.5 Conclusion

In this chapter, two semi-supervised methods for data embedding are proposed. For semi-supervised data embedding, the proposed methods utilize the label information from the labeled data and the manifold regularization (derived from sparsity preserving criterion) on both labeled and unlabeled training data. The FCSPE method can generate a linear projection for unseen data samples through a linear regression term in the optimal function.



(a) Yale

(b) ORL

(c) COIL-20

Fig. 7.4: Recognition rates variation as a function of different values of parameter μ and γ on Yale, ORL and COIL-20.

The experimental results on eight real image databases clearly demonstrate that the proposed methods can outperform the other competing embedding methods. Moreover, for misaligned faces, the proposed methods retained their superiority. The experimental results also give some hints for parameter selection in the proposed methods.

Part III

Conclusions

8

Conclusions and perspectives

Abstract

In this chapter, we summarize and conclude the developed work and discuss the advantages of the proposed methods as well as their limitations. Then we show some directions for future work. Finally, the publications extracted from the dissertation are listed.

Contents

8.1 Conclusions

The contributions of the present research work can be categorized into two main areas:

- Development of two graph construction methods via ℓ_2 and ℓ_1 minimization respectively which are based on data self-representativeness integrated by the constraints of Laplacian smoothness in the representation space.
- Development of two semi-supervised embedding methods.

In the case of graph construction, we integrate Laplacian smoothness as constraint with data self-representativeness. Take advantage of ℓ_2 coding, i.e. efficient and can easily receive optimal, we proposed a garph construction method based on data self-representativeness and Laplacian smoothness (S-RLS). Besides, we also introduced its kernelized variants and a direct solution. Since the ℓ_2 based representation can not promise the sparsity of the representation vector, we proposed two-phase SRLS (TPSRLS) to improve the sparsity of the affinity matrix.

Sparse graphs inherit advantages of sparse representation, sparsity and robust for noise. We proposed a sparse graph construction method with Laplacian smoothness (SGLS). We also introduce its kernelized variants and a direct solution.

In the case of semi-supervised learning, we proposed two semi-supervised methods for data embedding Constrained Sparsity Preserving Embedding (C-SPE) and flexible semi-supervised embedding method named Flexible Constrained Sparsity Preserving Embedding (FCSPE). The framework of CSPE does not provide a straightforward solution to the out-of-sample problem. Indeed, the regression is carried out as an extra step. With the flexible method, FCSPE, both the non-linear mapping and the regression are simultaneously estimated within one single framework.

8.2 Perspectives

1. Investigation on the efficient graph construction for large datasets. In graph construction, it will be time consuming when the dataset is very large since the size of affinity matrix is related to the number of samples. To improve the efficiency, subset selection methods can be used to reduce the dataset size. However graph construction over large dataset is still a big challenge.
2. Development of efficient label propagation methods and other semi-supervised classification methods. When graph is available, the next inferring task is also important. Label propagation methods or other semi-supervised classifier have some limitations such as most of them are transductive. One track is to develop some inductive semi-supervised classifier.
3. The integration of the technique of active learning into semi-supervised learning tasks. In real classification applications, it's normal that some samples are difficult to be classified. For instance, some data points near the decision boundary are difficult to be classified when a linear classifier is used. The technique of active learning can benefit such circumstance and improve the final performance.
4. The usage of other feature extraction methods in the experiments. In our experiments, we only use the gray level and LBP features of images as features. The other feature of images can also be a choice.
5. Different classifier can be used in semi-supervised embedding methods. we use nearest neighbor and SVM as classifier, the use of the other classifiers such as random forest, classifier based on neural network might be interesting to explore.
6. Development of a new framework which can obtain a non-linear mapping, a affinity graph and an approximate linear projection are estimated within one criterion.
7. Extension of the graph construction method to the case of multi-modal features.

8.3 Publications

The research work described produced some publications which are listed below.

International journal articles

1. L. Weng, F. Dornaika and Z. Jin, *Flexible constrained sparsity preserving embedding*, Pattern Recognition, Volume 60, December 2016, Pages 813-823.

2. L. Weng, F. Dornaika and Z. Jin, *Graph construction based on data self-representativeness and Laplacian smoothness*, Neurocomputing, Volume 207, September 2016, Pages 476-487.

3. L. Weng, F. Dornaika and Z. Jin, *Learning with Constrained Sparse Graph for Image Classification*, Submitted to Artificial Intelligence.

4. L. Weng, F. Dornaika and Z. Jin, *Structured Sparse Graphs Using Manifold constraints for visual data analysis*, Submitted to IEEE Trans. on Neural Networks and Learning Systems.

Book chapter

1. L. Weng, F. Dornaika and Z. Jin. *Constrained graph embedding based on Sparsity Preserving Projection*, Advances in Face Image Analysis: Theory and Applications, Bentham Science Publishers, 2015, Pages 23-38.

2. L. Weng, F. Dornaika and Z. Jin. *Constrained data self-representative graph construction*, semi-supervised Learning: Background, Applications and Future Directions, Nova Publisher, 2017.

3. L. Weng, F. Dornaika, and Z. Jin. Efficient graph construction through constrained data self-representativeness. International Conference on Artificial Neural Networks. ICANN, Alghero, Sardinia, Italy, 2017.

References

1. Jacob Abernethy, Olivier Chapelle, and Carlos Castillo, *Web spam identification through content and hyperlinks*, In proceedings of International Workshop on Adversarial Information Retrieval on the Web, ACM, 2008, pp. 41–44.
2. Suresh Balakrishnama and Aravind Ganapathiraju, *Linear discriminant analysis-a brief tutorial*, Institute for Signal and Information Processing **18** (1998), 1–9.
3. Shumeet Baluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly, *Video suggestion and discovery for youtube: taking random walks through the view graph*, In proceedings of International Conference on World Wide Web, ACM, 2008, pp. 895–904.
4. Mikhail Belkin and Partha Niyogi, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, In proceedings of Neural Information Processing Systems, vol. 14, 2001, pp. 585–591.
5. ———, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation **15** (2003), no. 6, 1373–1396.
6. Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, Journal of Machine Learning Research **7** (2006), 2399–2434.
7. Avrim Blum and Tom Mitchell, *Combining labeled and unlabeled data with co-training*, In proceedings of Conference on Computational Learning Theory, ACM, 1998, pp. 92–100.
8. Ingwer Borg and Patrick JF Groenen, *Modern multidimensional scaling: Theory and applications*, Springer Science & Business Media, 2005.
9. Leo Breiman, *Bagging predictors*, Machine learning **24** (1996), no. 2, 123–140.
10. Deng Cai, Xiaofei He, and Jiawei Han, *Semi-supervised discriminant analysis*, In proceedings of International Conference on Computer Vision, IEEE, 2007, pp. 1–7.
11. ———, *Spectral regression for efficient regularized subspace learning*, In proceedings of International Conference on Computer Vision, IEEE, 2007, pp. 1–8.
12. D. Calvetti and L. Reichel, *Application of ADI iterative methods to the restoration of noisy images*, SIAM Journal on Matrix Analysis and Applications **17** (1996), 165–186.

13. Gustavo Camps-Valls, Tatyana V Bandos Marsheva, and Dengyong Zhou, *Semi-supervised graph-based hyperspectral image classification*, IEEE Transactions on Geoscience and Remote Sensing **45** (2007), no. 10, 3044–3054.
14. Hakan Cevikalp, Jakob Verbeek, Frédéric Jurie, and Alexander Klaser, *Semi-supervised dimensionality reduction using pairwise equivalence constraints*, In proceedings of International Conference on Computer Vision Theory and Applications, vol. 1, INSTICC, 2008, pp. 489–496.
15. O. Chapelle, B. Scholkopf, and A. Zien, *Semi-supervised learning*, MIT Press, 2006.
16. Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, *A discussion of semi-supervised learning and transduction*, MIT Press, 2006.
17. Chun Chen, Lijun Zhang, Jiajun Bu, Can Wang, and Wei Chen, *Constrained laplacian eigenmap for dimensionality reduction*, Neurocomputing **73** (2010), no. 4, 951–958.
18. Hwann Tzong Chen, Huang Wei Chang, and Tyng Luh Liu, *Local discriminant embedding and its variants*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, IEEE, 2005, pp. 846–853.
19. S B Chen, Chris HQ Ding, and Bin Luo, *Similarity learning of manifold data*, IEEE Transactions on Cybernetics **99** (2014), 1–13.
20. Bin Cheng, Jianchao Yang, Shuicheng Yan, Yun Fu, and Thomas S Huang, *Learning with l1 graph for image analysis*, IEEE Transactions on Image Processing **19** (2010), no. 4, 858–866.
21. Yan Cui, Xiaodong Cai, and Zhong Jin, *Semi-supervised classification using sparse representation for cancer recurrence prediction*, In proceedings of IEEE International Workshop on Genomic Signal Processing and Statistics, 2013, pp. 102–105.
22. S.I. Daitch, J.A. Kelner, and D.A. Spielman, *Fitting a graph to vector data*, In proceedings of International Conference on Machine Learning, 2009, pp. 201–208.
23. Celso André R de Sousa, Solange O Rezende, and Gustavo EAPA Batista, *Influence of graph construction on semi-supervised learning*, Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 160–175.
24. Arthur P Dempster, Nan M Laird, and Donald B Rubin, *Maximum likelihood from incomplete data via the em algorithm*, Journal of the royal statistical society. Series B (methodological) (1977), 1–38.
25. F Dornaika and I Kamal Aldine, *Decremental sparse modeling representative selection for prototype selection*, Pattern Recognition **48** (2015), no. 11, 3714–3727.
26. F. Dornaika, A. Bosaghzadeh, and B. Raducanu, *Efficient graph construction for label propagation based multi-observation face recognition*, In proceedings of International Workshop on Human Behavior Understanding, 2013, pp. 124–135.
27. F Dornaika, A Bosaghzadeh, H Salmane, and Y Ruichek, *Graph-based semi-supervised learning with local binary patterns for holistic object categorization*, Expert Systems with Applications **41** (2014), no. 17, 7744–7753.
28. Axel Dreves, Francisco Facchinei, Christian Kanzow, and Simone Sagratella, *On the solution of the kkt conditions of generalized nash equilibrium problems*, SIAM Journal on Optimization **21** (2011), no. 3, 1082–1108.
29. Richard O Duda, Peter E Hart, and David G Stork, *Pattern classification*, John Wiley & Sons, 2012.

30. X. Fang, Y. Xu, X. Li, Z. Lai, and W.K. Wong, *Learning a nonnegative sparse graph for linear regression*, IEEE Transactions on Image Processing **24** (2015), no. 9, 2760–2771.
31. Yoav Freund and Robert E Schapire, *A desicion-theoretic generalization of on-line learning and an application to boosting*, In proceedings of European Conference on Computational Learning Theory, Springer, 1995, pp. 23–37.
32. C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang, *Deformed graph laplacian for semi-supervised learning*, IEEE Transactions on Neural Networks and Learning Systems **26** (2015), no. 10, 2261–2274.
33. Isabelle Guyon and André Elisseeff, *An introduction to variable and feature selection*, Journal of Machine Learning Research **3** (2003), 1157–1182.
34. Xiaofei He, Deng Cai, Shuicheng Yan, and Hong-Jiang Zhang, *Neighborhood preserving embedding*, In proceedings of IEEE International Conference on Computer Vision, vol. 2, IEEE, 2005, pp. 1208–1213.
35. Xiaofei He, Ming Ji, and Hujun Bao, *Graph embedding with constraints*, In proceedings of International Joint Conference on Artificial Intelligence, vol. 9, 2009, pp. 1065–1070.
36. Xiaofei He and Partha Niyogi, *Locality preserving projections*, In proceedings of Neural Information Processing Systems, vol. 16, 2003, pp. 234–241.
37. David W Hosmer Jr, *A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample*, Biometrics (1973), 761–770.
38. Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He, *Fast and accurate matrix completion via truncated nuclear norm regularization*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 9, 2117–2130.
39. Yi Huang, Dong Xu, and Feiping Nie, *Semi-supervised dimension reduction using trace ratio criterion*, IEEE Transactions on Neural Networks and Learning Systems **23** (2012), no. 3, 519–526.
40. A. Iosifidis, A. Tefas, and I. Pitas, *Graph embedded extreme learning machine*, IEEE Transactions on Cybernetics **46** (2016), no. 1, 311–324.
41. Tony Jebara, Jun Wang, and Shih Fu Chang, *Graph construction and b-matching for semi-supervised learning*, In proceedings of International Conference on Machine Learning, ACM, 2009, pp. 441–448.
42. Bryn Ll Jones, Eric C Kerrigan, and Jonathan F Morrison, *A modeling and filtering framework for the semi-discretised navier-stokes equations*, In proceedings of European Control Conference, IEEE, 2009, pp. 1215–1220.
43. Minyoung Kim and Fernando Torre, *Local minima embedding*, In proceedings of International Conference on Machine Learning, 2010, pp. 527–534.
44. Z. Lai, W. K. Wong, Z. Jin, J. Yang, and Y. Xu, *Sparse approximation to the eigensubspace for discrimination*, IEEE Transactions on Neural Networks and Learning Systems **23** (2012), no. 12, 1948–1960.
45. Z. Lai, W. K. Wong, Y. Xu, J. Yang, and D. Zhang, *Approximate orthogonal sparse embedding for dimensionality reduction*, IEEE Transactions on Neural Networks and Learning Systems **27** (2016), no. 4, 723–735.
46. Z. Lai, W. K. Wong, Y. Xu, C. Zhao, and M. Sun, *Sparse alignment for robust tensor learning*, IEEE Transactions on Neural Networks and Learning Systems **25** (2014), no. 10, 1779–1792.

47. Z. Lai, Y. Xu, Q. Chen, J. Yang, and D. Zhang, *Multilinear sparse principal component analysis*, IEEE Transactions on Neural Networks and Learning Systems **25** (2014), no. 10, 1942–1950.
48. Yongmin Li, Shaogang Gong, and Heather Liddell, *Kernel discriminant analysis*, ACM Transactions on Programming Languages and Systems **15** (1998), no. 5, 745–770.
49. Z. Li, Z. Lai, Y. Xu, J. Yang, and D. Zhang, *A locality-constrained and label embedding dictionary learning algorithm for image classification*, IEEE Transactions on Neural Networks and Learning Systems **PP** (2015), no. 99, 1–16.
50. Chien Liang Liu, Wen Hoar Hsaio, Chia Hoang Lee, and Fu Sheng Gou, *Semi-supervised linear discriminant clustering*, IEEE Transactions on Cybernetics **44** (2014), no. 7, 989–1000.
51. Wei Liu and Shih Fu Chang, *Robust multi-class transductive learning with graphs*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 381–388.
52. Wei Liu, Junfeng He, and Shih Fu Chang, *Large graph construction for scalable semi-supervised learning*, In proceedings of International Conference on Machine Learning, 2010, pp. 679–686.
53. Geoffrey J McLachlan and S Ganesalingam, *Updating a discriminant function on the basis of unclassified data*, Communications in Statistics-Simulation and Computation **11** (1982), no. 6, 753–767.
54. Feiping Nie, Hua Wang, Heng Huang, and Chris Ding, *Adaptive loss minimization for semi-supervised elastic embedding*, In proceedings of International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1565–1571.
55. Feiping Nie, Shiming Xiang, Yangqing Jia, and Changshui Zhang, *Semi-supervised orthogonal discriminant analysis via label propagation*, Pattern Recognition **42** (2009), no. 11, 2615–2627.
56. Feiping Nie, Dong Xu, Xuelong Li, and Shiming Xiang, *Semi-supervised dimensionality reduction and classification through virtual label regression*, IEEE Transactions on Systems, Man, and Cybernetics, Part B **41** (2011), no. 3, 675–685.
57. Feiping Nie, Dong Xu, Ivor Wai Hung Tsang, and Changshui Zhang, *Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction*, IEEE Transactions on Image Processing **19** (2010), no. 7, 1921–1932.
58. Kamal Paul Nigam, *Using unlabeled data to improve text classification*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, May 2001.
59. Partha Niyogi, *Manifold regularization and semi-supervised learning: some theoretical analyses.*, Journal of Machine Learning Research **14** (2013), no. 1, 1229–1250.
60. T. Ojala, M. Pietikäinen, and T. Maenpaa, *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002), 971–987.
61. Matan Orbach and Koby Crammer, *Graph-based transduction with confidence*, In proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2012, pp. 323–338.
62. Feng Pan, Jiandong Wang, and Xiaohui Lin, *Local margin based semi-supervised discriminant embedding for visual recognition*, Neurocomputing **74** (2011), no. 5, 812–819.

63. P. Qian, F. Chung, S. Wang, and Z. Deng, *Fast graph-based relaxed clustering for large data sets using minimal enclosing ball*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **42** (2012), no. 3, 672–687.
64. Lishan Qiao, Songcan Chen, and Xiaoyang Tan, *Sparsity preserving discriminant analysis for single training image face recognition*, Pattern Recognition Letters **31** (2010), no. 5, 422 – 429.
65. Lishan Qiao, Songcan Chen, and Xiaoyang Tan, *Sparsity preserving projections with applications to face recognition*, Pattern Recognition **43** (2010), no. 1, 331–341.
66. Sam T. Roweis and Lawrence K. Saul, *Nonlinear dimensionality reduction by locally linear embedding*, Science **290** (2000), no. 5500, 2323–2326.
67. Bernhard Schölkopf, Alexander Smola, and Klaus Robert Müller, *Kernel principal component analysis*, In proceedings of International Conference on Artificial Neural Networks, Springer, 1997, pp. 583–588.
68. H Scudder, *Probability of error of some adaptive pattern-recognition machines*, IEEE Transactions on Information Theory **11** (1965), no. 3, 363–371.
69. Amir Shahzad, Bryn Ll Jones, Eric C Kerrigan, and George A Constantinides, *An efficient algorithm for the solution of a coupled sylvester equation appearing in descriptor systems*, Automatica **47** (2011), no. 1, 244–248.
70. Blake Shaw and Tony Jebara, *Structure preserving embedding*, In proceedings of International Conference on Machine Learning, ACM, 2009, pp. 937–944.
71. Qinfeng Shi, Anders Eriksson, Anton Van Den Hengel, and Chunhua Shen, *Is face recognition really a compressive sensing problem?*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 553–560.
72. Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin, *Beyond the point cloud: from transductive to semi-supervised learning*, In proceedings of International Conference on Machine learning, ACM, 2005, pp. 824–831.
73. Vikas Sindhwani, Partha Niyogi, Mikhail Belkin, and Sathiya Keerthi, *Linear manifold regularization for large scale semi-supervised learning*, In proceedings of International Conference on Machine Learning Workshop on Learning with Partially Classified Training Data, vol. 28, 2005.
74. C. Sousa, S. Rezende, and G. Batista, *Influence of graph construction on semi-supervised learning*, In proceedings of European Conferene on Machine Learning, 2013, pp. 160–175.
75. Amarnag Subramanya and Jeff Bilmes, *Semi-supervised learning with measure propagation*, Journal of Machine Learning Research **12** (2011), 3311–3370.
76. Amarnag Subramanya and Partha Pratim Talukdar, *Graph-based semi-supervised learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning **8** (2014), no. 4, 1–125.
77. Masashi Sugiyama, Tsuyoshi Idé, Shinichi Nakajima, and Jun Sese, *Semi-supervised local fisher discriminant analysis for dimensionality reduction*, Machine learning **78** (2010), no. 1-2, 35–61.
78. Shiliang Sun, Zakria Hussain, and John Shawe Taylor, *Manifold-preserving graph reduction for sparse semi-supervised learning*, Neurocomputing **124** (2014), 13–21.
79. V. Takala, T. Ahonen, and M. Pietikäinen, *Block-based methods for image retrieval using local binary patterns*, In proceedings of Scandinavian Conference on Image Analysis, 2005, pp. 882–891.

80. Partha Pratim Talukdar and Koby Crammer, *New regularized algorithms for transductive learning*, In proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2009, pp. 442–457.
81. J. Tang, L. Shao, X. Li, and K. Lu, *A local structural descriptor for image matching via normalized graph laplacian embedding*, IEEE Transactions on Cybernetics **46** (2016), no. 2, 410–420.
82. Joshua B Tenenbaum, Vin De Silva, and John C Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science **290** (2000), no. 5500, 2319–2323.
83. Isaac Triguero, Salvador García, and Francisco Herrera, *Seg-ssc: A framework based on synthetic examples generation for self-labeled semi-supervised classification*, IEEE Transactions on Cybernetics **45** (2015), no. 4, 622–634.
84. Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik, *Dimensionality reduction: a comparative review*, J Mach Learn Res **10** (2009), 66–71.
85. Vladimir Naumovich Vapnik and Vlamimir Vapnik, *Statistical learning theory*, vol. 1, Wiley, 1998.
86. Changhu Wang, Shuicheng Yan, Lei Zhang, and Hong-Jiang Zhang, *Multi-label sparse coding for automatic image annotation*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 1643–1650.
87. J. Wang, C. Lu, M. Wang, P. Li, S. Yan, and X. Hu, *Robust face recognition via adaptive sparse representation*, IEEE Transactions on Cybernetics **44** (2014), no. 12, 2368–2378.
88. Jingdong Wang, Fei Wang, Changshui Zhang, Helen C Shen, and Long Quan, *Linear neighborhood propagation and its applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009), no. 9, 1600–1615.
89. Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong, *Locality-constrained linear coding for image classification*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3360–3367.
90. Jadoon Waqas, Zhang Yi, and Lei Zhang, *Collaborative neighbor representation based classification using l2-minimization approach*, Pattern Recognition Letters **34** (2013), no. 2, 201–208.
91. Jia Wei and Hong Peng, *Neighbourhood preserving based semi-supervised dimensionality reduction*, Electronics Letters **44** (2008), no. 20, 1190–1192.
92. Kilian Q Weinberger and Lawrence K Saul, *Unsupervised learning of image manifolds by semidefinite programming*, International Journal of Computer Vision **70** (2006), no. 1, 77–90.
93. Svante Wold, Kim Esbensen, and Paul Geladi, *Principal component analysis*, Chemometrics and Intelligent Laboratory Systems **2** (1987), no. 1-3, 37–52.
94. John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma, *Robust face recognition via sparse representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009), no. 2, 210–227.
95. Zenglin Xu, Irwin King, Michael Rung Tsong Lyu, and Rong Jin, *Discriminative semi-supervised feature selection via manifold regularization*, IEEE Transactions on Neural Networks **21** (2010), no. 7, 1033–1047.
96. J. Xuan, J. Lu, G. Zhang, and X. Luo, *Topic model for graph mining*, IEEE Transactions on Cybernetics **45** (2015), no. 12, 2792–2803.
97. Shuicheng Yan and Huan Wang, *Semi-supervised learning by sparse representation*, In proceedings of International Conference on Data Mining, SIAM, 2009, pp. 792–801.

98. Bo Yang and Songcan Chen, *A comparative study on local binary pattern (lbp) based face recognition: Lbp histogram versus lbp image*, Neurocomputing **120** (2013), no. 23, 365–379, Image Feature Detection and Description.
99. Shuyuan Yang, Xiuxiu Wang, Min Wang, Yue Han, and Licheng Jiao, *Semi-supervised low-rank representation graph for pattern recognition*, IET Image Processing **7** (2013), no. 2, 131–136.
100. Wuyi Yang, Shuwu Zhang, and Wei Liang, *A graph based subspace semi-supervised learning framework for dimensionality reduction*, In proceedings of European Conference on Computer Vision, Springer, 2008, pp. 664–677.
101. Guoxian Yu, Guoji Zhang, Carlotta Domeniconi, Zhiwen Yu, and Jane You, *Semi-supervised classification based on random subspace dimensionality reduction*, Pattern Recognition **45** (2012), no. 3, 1119–1135.
102. Daoqiang Zhang, Songcan Chen, Zhi Hua Zhou, and Qiang Yang, *Constraint projections for ensemble learning.*, In proceedings of Association for the Advancement of Artificial Intelligence, 2008, pp. 758–763.
103. Daoqiang Zhang, Zhi Hua Zhou, and Songcan Chen, *Semi-supervised dimensionality reduction*, In proceedings of International Conference on Data Mining, SIAM, 2007, pp. 629–634.
104. L. Zhang, Y. Gao, C. Hong, Y. Feng, J. Zhu, and D. Cai, *Feature correlation hypergraph: Exploiting high-order potentials for multimodal recognition*, IEEE Transactions on Cybernetics **44** (2014), no. 8, 1408–1419.
105. Lei Zhang, Meng Yang, and Xiangchu Feng, *Sparse representation or collaborative representation: Which helps face recognition?*, In proceedings of IEEE International Conference on Computer vision, IEEE, 2011, pp. 471–478.
106. Tong Zhang, Alexandrin Popescul, and Byron Dom, *Linear prediction models with graph regularization for web-page categorization*, In proceedings of Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 821–826.
107. Tongtao Zhang, Rongrong Ji, Wei Liu, Dacheng Tao, and Gang Hua, *Semi-supervised learning with manifold fitted graphs*, In proceedings of International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1896–1902.
108. Yan-Ming Zhang, Kaizhu Huang, Xinwen Hou, and Cheng-Lin Liu, *Learning locality preserving graph from data*, IEEE Transactions on Cybernetics **44** (2014), no. 11, 2088–2098.
109. Yu Zhang and Dit Yan Yeung, *Semi-supervised discriminant analysis using robust path-based similarity*, In proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.
110. Zhen Yue Zhang and Hong Yuan Zha, *Principal manifolds and nonlinear dimensionality reduction via tangent space alignment*, Journal of Shanghai University (English Edition) **8** (2004), no. 4, 406–424.
111. Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf, *Learning with local and global consistency*, Advances in Neural Information Processing Systems **16** (2004), no. 16, 321–328.
112. Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf, *Learning from labeled and unlabeled data on a directed graph*, In proceedings of International Conference on Machine Learning, ACM, 2005, pp. 1036–1043.
113. Xiaojin Zhu, *Semi-supervised learning*, Encyclopedia of Machine Learning, Springer, 2010, pp. 892–897.
114. Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al., *Semi-supervised learning using gaussian fields and harmonic functions*, In proceedings of International Conference on Machine Learning, vol. 3, 2003, pp. 912–919.